MÁRCIO RODRIGO BRAZ

MÉTRICAS DE SOFTWARE BASEADAS EM CASOS DE USO E TEORIA *FUZZY*

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática, Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientadora: Prof.ª Silvia Regina Vergilio

CURITIBA
2004

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Márcio Rodrigo Braz, avaliamos o trabalho intitulado, "*METRICAS DE SOFTWARE BASEADAS EM CASOS DE USO E TEORIA FUZZY* ", cuja defesa foi realizada no dia 26 de abril de 2004, às quatorze horas, na sala de reuniões do CCE do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 26 de abril de 2004.

Profª. Dra. Silvia Regina Vergílio
DINF/UFPR – Orientadora

Profª. Dra. Renata Pontin de Mattos Fortes
USP - Membro Externo

Prof. Dr. Hélio Pedrini
DINF/UFPR – Membro Interno

A Deus (razão de tudo),
ao Seu João (meu pai),
à Dona Vitória (minha mãe),
e à Alaíne (minha esposa).

Agradecimentos

Há dois anos, já fazia outros dois anos que eu estava namorando à distância a Alaíne. Nossos planos previam que eu retornasse a Foz assim que terminasse a graduação. No entanto, isso não ocorreu; acabei ficando em Curitiba para cursar o Mestrado, o que significou mais dois anos de afastamento (até que ela se formasse). Por isso, eu devo agradecer muito a ela, pois embora contrariada, deu-me apoio incondicional nesta difícil decisão. Hoje, fico feliz por ter cursado o mestrado, mas ainda mais feliz por ter me casado com ela.

Agradeço à minha querida mãe, que sempre manifestou muita preocupação com todas aquelas siglas e cálculos cada vez que resolvia folhear um livro meu. Ela também merece ser mais agradecida, pois o que perde de lágrimas a cada vez que volto a Curitiba não é fácil! Também agradeço ao meu pai por todo o suporte prestado enquanto eu estava em Curitiba e por ter sido um dos primeiros a me incentivar a fazer este curso.

Agradeço a toda a minha família, meus irmãos Eliane e Edson; meus cunhados e cunhadas, Valter, Marcela, Léo, Flá e Lú (da "casa de praia"em Joinville); meus sogros, Dona Lucia e Seu Toninho; e aos meus sobrinhos, Gê, João Vítor e Lucas.

Agradeço aos colegas do Mestrado na UFPR: Cláudia, Evandro, Gustavo, Marcelo, Lucélia, entre outros;

Agradeço à Silvia, com quem aprendi a gostar muito da pesquisa acadêmica. Agradeço por toda a orientação e compreensão, principalmente nos momentos em que tive que conciliar o Mestrado, o trabalho no TJ e as aulas na FAEC. Também agradeço-a por ter visualizado o meu interesse por uma área um pouco diferente da que estava orientando na época (Teste de Software) e, assim, facilitando toda a elaboração da dissertação.

Agradeço ao professor Hélio e à Letícia (ambos da UFPR), pelas contribuições durante a defesa da proposta e à professora Renata da USP, pela participação e sugestões durante a defesa da dissertação.

Agradeço ao coordenador do Mestrado e meu professor na graduação, Direne, sempre atencioso e disposto a ajudar, seja às 8 horas de segunda ou às 22 horas de sexta.

Agradeço aos meus chefes e ex-chefes, já que, sem a conivência e compreensão deles, não teria conseguido cursar o Mestrado: Sandro da Datran, que também ajudou-me no estudo de caso; à Sandra da ALL, sempre muito gentil e companheira; ao André da ALL; e à Inês do TJ.

Quero agradecer a todos os meus colegas e amigos do TJ e, em particular, aos colegas de equipe: Alberto, Gustavo e Wagner, pela paciência que tiveram comigo durante as etapas mais difíceis do curso, que retiravam minha concentração do ambiente de trabalho.

Também agradeço a alguns dos meus melhores amigos: Hidalgo e Fábio (amigões, sempre por perto), Juliano, Eduardo e Victor (foi duro manter aquele ap. na André de Barros em pé), Gigio e Lauro (grandes amigos desde o tempo da MPS).

Aos amigos da Datran, que comemoraram comigo no Bar do Bosque quando passei no Poscomp: Almir (também passamos por várias na fatídica Sala Argentina) e Rodrigo Othávio. Aos amigos da ALL, com os quais passei muitas madrugadas trabalhando para o trens não pararem: Edinho, Ricardo, Roberto, Anderson. Agradeço especialmente ao Arthur, amigo da Datran e da ALL, que também me ajudou no estudo de caso, inclusive, foi até o TJ levar material para mim na véspera de seu casamento.

A todo o pessoal da FAEC: meus alunos, colegas professores e funcionários, todos muito atenciosos e prestativos. Agradeço em especial à professora Alexsandra e à diretora Marystela, que foram muito compreensivas quando precisei do afastamento para elaboração da dissertação.

Enfim, quero agradecer a todos os meus amigos, colegas e familiares, que só não foram citados porque não posso fazer desse breve agradecimento uma segunda dissertação. A todos vocês, muito obrigado!

"Dizem que sou louco. Por pensar assim.
Se eu sou muito louco. Por eu ser feliz.
Mais louco é quem me diz.
E não é feliz, não é feliz..."

Rita Lee e Arnaldo Batista

Resumo

Estimar esforço e custo de software é uma atividade muito importante que envolve elementos com um alto grau de incerteza. Diferentes métodos, em sua maioria baseados em Linhas de Código (Lines of Code-LOC) ou Pontos por Função (Function Points-FP), ajudam os gerentes nesta atividade, presente nos estágios iniciais do desenvolvimento de um software. Entretanto, as tecnologias de orientação a objetos trouxeram novos modelos para a especificação de software, tal como o modelo de Casos de Uso (Use Case-UC), amplamente utilizado na maioria das organizações. Conseqüentemente, foram propostas extensões para as métricas tradicionais; entre elas, a métrica Pontos por Caso de Uso (Use Case Points-UCP). O UCP considera aspectos funcionais de um UC, mas apresenta algumas limitações principalmente associadas à granularidade do UC. Por exemplo, o modelo tem apenas três categorias para classificar a complexidade de um UC. Devido a isso, o UCP pode, por exemplo, classificar na mesma categoria UCs grandes e aqueles que são muito grandes. Para superar essas limitações, este trabalho introduz duas métricas também baseadas em UCs. A primeira, chamada Pontos por Tamanho de Caso de Uso (Use Case Size Points-USP), considera as estruturas internas do UC, medindo mais adequadamente a sua funcionalidade. A segunda, denominada Pontos por Tamanho de Caso de Uso Fuzzy (Fuzzy Use Case Size Points-USPF), considera conceitos da Teoria dos Conjuntos *Fuzzy* para criar classificações graduais que se adaptam melhor à incerteza. As duas métricas foram utilizadas em um estudo de caso com um projeto real e foram comparadas com as métricas UCP e FP. Os resultados obtidos demonstram a aplicabilidade e algumas vantagens das métricas propostas. Para automatizar a coleta destas métricas também é proposta uma notação para escrever UCs no formato de documentos XML, estes documentos são utilizados por um protótipo que foi desenvolvido para efetuar os cálculos necessários.

Abstract

Estimating software effort and costs is a very important activity that includes many uncertain elements. Different methods, mainly based on Lines of Codes (LOC) or Function Points (FP), help the managers in this early activity of the software development. However, object-oriented technologies brought new models for software specification, such as the Use Case (UC) model, widely used in most organizations. Consequently, extensions to the traditional metrics were proposed; among them, the metric Use Case Points (UCP). The UCP considers the functional aspects of an UC, but presents some limitations mainly related to the granularity of the UC. For example, it has only three categories of complexity for the UCs. Because of this, the UCP, could classify in the same category large and huge UCs. To overcome these limitations, this work introduces two metrics, also based on UCs. The first one, named USP (Use Case Size Points), considers the internal structures of the UC and better captures its functionality. The second one, named USPF (USP Fuzzy), considers concepts of the *Fuzzy Set Theory* to create gradual classifications that better deal with uncertainty. Both metrics were used in a case study with a real project and compared with the metrics UCP and FP. The obtained results show the applicability and some advantages of the proposed metrics. To automate the collect of these metrics is also proposed a notation to write UCs in the format of XML documents, these documents are used by a prototype developed to perform the needed calculations.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A atividade de gerenciamento de projetos de software é particularmente difícil nos estágios iniciais do desenvolvimento de um sistema. Isto é devido à carência de informações e subsídios para a tomada de decisões.

Diante desse dilema imposto aos gerentes de projeto, a atividade de predição de custo e tamanho de software desempenha papel fundamental. O processo de estimativa, no entanto, só pode ser realizado com a definição de uma métrica capaz de mensurar adequadamente o software a ser desenvolvido.

Tradicionalmente, os sistemas de software são medidos através de duas métricas, Linhas de Código (Lines of Code-LOC) [36] e Pontos por Função (Function Points-FP) [2], ambas possuem vantagens e desvantagens. A métrica LOC é de fácil automatização e objetiva, porém é bastante influenciada pela linguagem de programação utilizada. Além disso, ela pode ser coletada somente nas fases finais do desenvolvimento. A análise baseada em FP (Function Points Analysis-FPA) é provavelmente mais utilizada pela comunidade e, ao contrário da LOC, permite uma certa independência com relação à linguagem e paradigma utilizados, pois mede a funcionalidade presente nos requisitos de um software. A FPA apresenta como desvantagens: o emprego de subjetividade na análise e a necessidade de certos elementos que, freqüentemente, não estão disponíveis nos estágios iniciais de um projeto. A subjetividade empregada e a ausência de informações introduz um alto grau de incerteza.

Para contornar esses problemas algumas extensões à FPA foram propostas [21, 48]. Estas extensões utilizam a teoria *Fuzzy* para permitir a introdução de incerteza sobre a pertinência de elementos em determinados conjuntos [23]. A teoria *Fuzzy* tem sido aplicada com bastante sucesso em diversas áreas do conhecimento.

Outro ponto a ser considerado é que muitas companhias introduziram ori-

entação a objeto em seus ambientes de desenvolvimento e as métricas existentes necessitam alguma adaptação para serem utilizadas nesse contexto. Na literatura, encontram-se alguns trabalhos que incluem esforços nessa direção [15, 37, 43], principalmente visando estender a FPA para o software orientado a objeto.

A utilização das métricas convencionais mencionadas acima, para a mensuração de sistemas orientados a objeto, exige a transposição de um abismo filosófico existente entre os paradigmas. Além disso, a utilização da FPA nem sempre é possível. Por exemplo, o modelo de casos de uso (Use Case - UC) [20] tem sido amplamente utilizado para a especificação de requisitos de sistemas orientados a objeto. Os UCs representam aspectos funcionais dos sistemas e são muito úteis nas fases iniciais do software, onde estimações de esforço e custo devem ser realizadas. A FPA, entretanto, requer informações que não estão representadas nesse modelo e nesta fase do ciclo de vida do software.

Considerando os aspectos descritos, uma métrica chamada Pontos por Caso de Uso (Use Case Points - UCP) tornou-se muito conhecida e popular [3, 31]. O UCP foi criado por Gustav Karner [22] e utiliza o modelo de UCs para estimar o tamanho de sistemas orientados a objeto.

Embora o UCP seja uma métrica mais adequada para medir UCs, ela ainda apresenta algumas limitações que podem reduzir a precisão das estimativas geradas, por exemplo, o UCP não considera as estruturas internas do UC na definição de seu tamanho, tornando a avaliação superficial e altamente subjetiva. Um UC independentemente da quantidade de seções e/ou de seu conteúdo só pode ser classificado em três categorias (simples, médio ou complexo), limitando as possibilidades de quantificação da funcionalidade presente no mesmo.

1.1 Motivação

Dado o contexo exposto acima, resume-se os seguintes itens que se constituem motivação para o trabalho aqui descrito.

- Estimativas do tamanho de software em fases iniciais do desenvolvimento são fundamentais para se poder prever custo e esforço e auxiliar os gerentes de software na tomada de decisões;
- Métricas tradicionais são de difícil aplicação para estimar software orientado a objetos, visto que elas não levam em consideração os aspectos representados pelos principais modelos criados nas fases iniciais do desenvolvimento utilizando esse paradigma;

- O modelo de UC tem sido amplamente utilizado pela comunidade de desenvolvimento de software, o que tornou a métrica UCP, proposta para medir especificamente UCs, bastante promissora;
- A métrica UCP, apresenta deficiências, relacionadas principalmente à granularidade do UC, que podem prejudicar o processo de medição;
- A criação de um modelo simples como o UCP, mas que gere melhores estimativas, pode vir a ter uma boa aceitação;
- A Teoria dos Conjuntos *Fuzzy* tem sido utilizada com bastante sucesso para se estimar esforço juntamente com métricas tradicionais. Isso motiva a utilização desses conceitos em métricas baseadas em UCs; e
- A necessidade de que a coleta de métricas e estimativas seja feita de maneira automática para tornar o planejamento mais rápido e eficiente possível.

1.2 Objetivos do Trabalho

Esse trabalho tem como objetivo introduzir duas novas métricas de tamanho de software baseadas em UCs: USP (Use Case Size Points) e USPF (Fuzzy Use Case Size Points). Elas têm como objetivo reduzir as principais limitações da métrica UCP.

A métrica USP considera outros aspectos de um UC, descritos em uma notação expandida, tais como: número de cenários, pré e pós condições, etc. Esses aspectos capturam melhor a funcionalidade dos UCs. A métrica USPF considera conceitos da teoria dos conjuntos *Fuzzy* para reduzir a influência humana na classificação dos elementos de um UC, permitindo dessa maneira incerteza na estimação.

A automatização do processo de coleta das métricas é outro objetivo a ser alcançado, pois o sucesso de uma métrica depende não apenas de sua eficiência, mas também da facilidade de aplicação. Sendo assim, este trabalho propõe uma notação para a escrita de UCs em um formato estruturado e relativamente formal, que utiliza a linguagem XML. A notação em conjunto com um protótipo desenvolvido permite a aplicação automática das métricas propostas.

Para validar e testar a viabilidade das novas métricas, um estudo de caso com um projeto real se fez necessário, onde além de avaliadas, as métricas foram comparadas com a métrica FP, uma das métricas mais utilizadas pela comunidade de software. O estudo permitiu ainda a avaliação das possibilidades de automatização através do uso da notação e do protótipo construído.

1.3 Organização

Este trabalho está organizado da seguinte maneira: o Capítulo 2 introduz o modelo de Casos de Uso e conceitos relacionados. O Capítulo 3 descreve a Teoria dos Conjuntos *Fuzzy* e também a Lógica *Fuzzy*. O Capítulo 4, apresenta as principais métricas para a mensuração de software orientado a objetos, Análise de Pontos por Função e Pontos por Caso de Uso, outras métricas e extensões das tradicionais também são apresentadas; esse capítulo se encerra com a apresentação da Análise de Pontos por Função Fuzzy. O Capítulo 5 apresenta as duas métricas propostas, USP e USPF. A notação para escrita de UCs em XML e o protótipo utilizado na automatização da coleta da métrica são apresentados no Capítulo 6. Na seqüência, o Capítulo 7 descreve com detalhes o estudo de caso realizado. O Capítulo 8 conclui o trabalho e traz sugestões para trabalhos futuros.

Capítulo 2

Casos de Uso

Informalmente, um Caso de Uso (UC - Use Case) é uma história de utilização de um sistema para o cumprimento de objetivos [26], sejam estes objetivos dos usuários ou de outros sistemas.

Os usuários e/ou outros sistemas que interagem com o sistema em questão são denominados atores. A seguir um exemplo simples de UC:

> *Efetuando saque de dinheiro: o cliente do banco chega até o caixa automático desejando sacar uma determinada quantia. O cliente informa seus dados. O sistema faz a validação e autenticação das informações e fornece as opções ao cliente. O cliente então solicita a operação de saque, informando o valor a sacar. O sistema entrega o dinheiro ao cliente, registrando o valor do saque efetuado e abatendo este valor do saldo do cliente.*

Embora essa definição simplificada seja correta, faz-se necessária a avaliação de outros aspectos para uma compreensão completa e adequada do que são os UCs, os quais são apresentados nas seções seguintes.

2.1 Requisitos de Software

Os requisitos de um software definem as características desejáveis do sistema sendo desenvolvido. A tarefa de especificação de requisitos visa traduzir os desejos dos usuários de sistemas computacionais em documentos e diagramas explicativos, reduzindo ambigüidades e confusões que podem ser geradas pela

informalidade. Existem muitos modelos que visam à descrição de funcionalidades de um sistema de informação, alguns mais formais e sofisticados e outros mais simples. Eis as principais características necessárias a um método de especificação [39]:

- Tratamento de complexidade advinda de um grande número de informações;
- Ser legível pelo cliente do sistema, de forma a garantir que o sistema certo está sendo desenvolvido; e
- Ser robusto o suficiente para dar apoio às alterações, que certamente virão com o decorrer do tempo.

O UC é um modelo para especificação de requisitos, é uma documentação simples, relativamente formal e bastante útil.

2.2 Breve Histórico

A introdução do UC por Jacobson et al [20] veio suprir deficiências apresentadas pela maioria dos métodos de especificação existentes que, ora deixavam o usuário de lado, ora não auxiliavam o analista e o projetista em suas tarefas. Os UCs de Jacobson foram imediatamente aceitos devido a sua simplicidade e utilidade.

O próximo passo na evolução da técnica foi dado por Alistair Cockburn [9], que apresenta em seu livro, maneiras compreensíveis e inteligentes de se escrever casos de uso, minimizando a confusão gerada pela falta de uma notação específica que determine como escrever, o que escrever e como estruturar um UC.

2.3 Definições

O UC é utilizado para definir o comportamento de um sistema ou de outra entidade semântica, sem revelar a estrutura interna do sistema ou entidade [33]. Cada UC especifica uma seqüência de ações (incluindo variações) que o sistema pode executar durante uma interação com os usuários do sistema.

Embora estejam definidos pela Linguagem de Modelagem Unificada (Unified Modeling Language - UML) [33], a principal notação utilizada na modelagem de sistemas orientados a objeto, a utilização dos UCs não se limita aos softwares desenvolvidos sob tal paradigma. Os UCs pertencem à fase de extração e especificação de requisitos, e nesta fase a forma de construção

do software e suas tecnologias subjacentes não é, e nem deve ser conhecida. Portanto, também é possível aplicar os modelos de UC em qualquer projeto não orientado a objetos.

2.4 Elementos de um Caso de Uso

2.4.1 Ator

Um ator é um usuário ou sistema externo que interage com o sistema sendo especificado. Um ator não é uma pessoa em particular, mas sim o papel que essa pessoa desempenha em um dado momento de interação com o sistema. Uma mesma pessoa pode assumir diferentes papéis, ex: um gerente pode em um UC atuar como um simples funcionário desejando imprimir seu contracheque e em outro UC pode atuar como o gerente que deseja fiscalizar seus subordinados.

Os atores podem ser classificados em primários ou secundários. Os atores primários são aqueles para os quais o sistema foi desenvolvido. Os atores secundários dão suporte à utilização do sistema. Ex: Ator primário: cliente do banco que deseja sacar dinheiro. Ator secundário: operador que efetua o reabastecimento dos caixas automáticos.

2.4.2 Cenário

Basicamente, um cenário é uma seqüência específica de interações entre atores e o sistema. Dessa forma, um UC é uma coleção de cenários de sucesso e de insucesso, os quais descrevem a utilização do sistema pelos atores.

Os cenários de sucesso normalmente indicam o caminho principal do UC e os caminhos alternativos em que o usuário consegue o atendimento de sua meta. Por outro lado, os cenários de insucesso registram os caminhos percorridos pelo usuário que por algum motivo o levaram a não atingir seu objetivo.

Eis alguns exemplos:

- Ex 1: Cenário de sucesso: o cliente chega até o caixa automático, informa o valor a sacar, o sistema processa o saque e entrega o dinheiro.
- Ex 2: Cenário alternativo: o cliente informou uma senha inválida, o sistema deve solicitar a senha novamente, o cliente repete o preenchimento da senha e o caixa valida a senha. Em seguida, o cliente informa o valor a sacar, o sistema processa o saque e entrega o dinheiro.

- Ex:3 Cenário de insucesso (Exceção): o cliente informou uma senha inválida por três vezes, nesse caso o sistema bloqueia o cartão impedindo toda e qualquer operação.

2.5 Diagramas

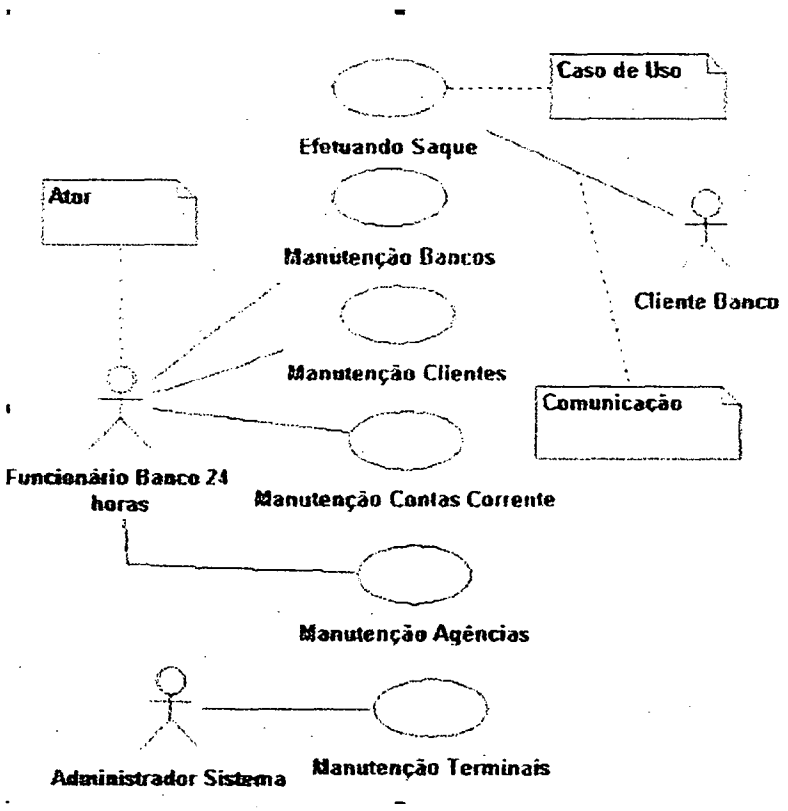

Figura 2.1: Diagrama de Casos de Uso

A interação entre os UCs e os atores de um sistema pode ser representada por diagramas, a UML define uma notação específica para a composição de tais artefatos.

Os diagramas são importantes para fornecer uma visão geral e abrangente do sistema, no entanto, alguns autores defendem um enfoque maior na escrita dos UCs do que em sua diagramação [9, 26].

A Figura 2.1 apresenta um exemplo de diagrama de UC. O exemplo contempla um sistema bancário simples. Os bonecos representam os atores (Funcionário, Cliente, Administrador), as elipses representam os UCs do

sistema(Efetuando Saque, Manutenção de Clientes, etc), e as setas que ligam os atores aos UCs indicam a existência de interação (comunicação) entre eles.

2.6 Relacionamentos entre UCs

A construção de um sistema real implica na identificação de uma grande quantidade de UCs e cenários. Para que se tenha uma manutenção adequada de um conjunto de UCs, o modelo deve ser robusto o suficiente para permitir modificações. Também é necessário que o modelo favoreça o reuso de soluções. Visando ao atendimento dessas necessidades, a UML define dois tipos de relacionamentos entre UCs: "include" (inclusão) e "extends" (extensão).

2.6.1 Relacionamento de Inclusão

"Analisando-se o conjunto de casos de uso já descritos, freqüentemente descobre-se que vários deles utilizam partes de roteiros comuns, ao retirarmos a parte comum destes casos e colocá-la em um caso separado (fatorando ou colocando em evidência) estaremos simplificando em muito o nosso modelo, pois agora teremos esta parte da rotina escrita em apenas um único lugar" [39].

Então, um UC pode fazer uso de outro caso através desta construção, evitando-se assim a re-especificação de funcionalidades já documentadas. Ex: O UC "efetuando saque" "inclui" o UC "validando cartão e senha", evitando detalhar novamente esta operação.

2.6.2 Relacionamento de Extensão

Para permitir a evolução do modelo de UCs, existe o relacionamento de extensão. Com esta associação torna-se possível adicionar funcionalidades aos UCs já existentes sem alteração do conteúdo do caso original, que já estava suficientemente descrito. A seguir um exemplo de relacionamento de extensão:

Ex: Suponha o UC "efetuando saque de dinheiro", digamos que é desejável criar uma nova funcionalidade que permita ao cliente em algumas situações efetuar o saque e, na mesma operação receber um comprovante do saque que informe o valor sacado e o saldo restante. Nesse caso poderíamos criar uma extensão do UC principal, da seguinte forma:

> *Efetuando saque de dinheiro com comprovante: Esse UC é inserido após o término do UC "efetuando saque de dinheiro" quando o cliente escolheu a opção "saque com comprovante". Após a execução do UC anterior, o sistema emite um comprovante que indica o valor sacado pelo cliente e o saldo restante em sua conta corrente.*

Os relacionamentos de inclusão e extensão possuem sua própria notação gráfica, como pode ser observado na Figura 2.2

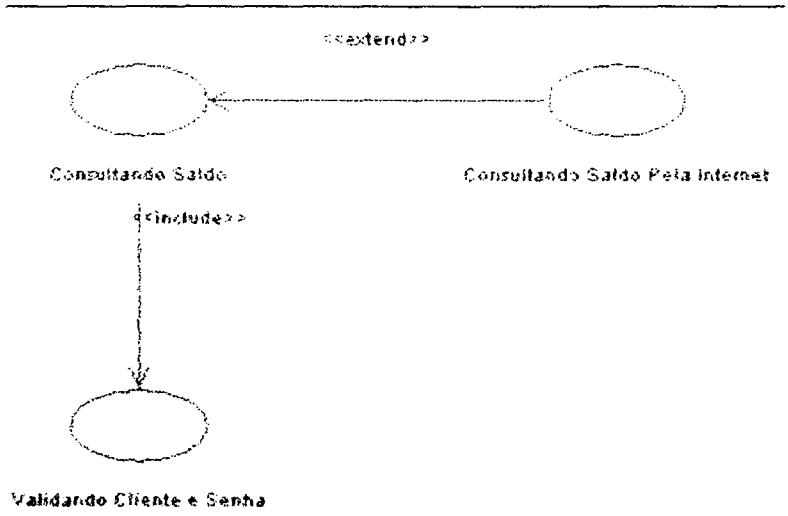

Figura 2.2: Exemplo de Relacionamento entre Casos de Uso

2.7 Roteiro de Criação de UCs [26, 39]

Visando facilitar a construção dos UCs de um sistema, é desejável a utilização de um roteiro que defina os passos necessários à execução de tal atividade. O roteiro transcrito aqui foi construído a partir de roteiros similares, encontrados em [26, 39].

2.7.1 Identificando Limites

Primeiramente, defina os limites da aplicação. A aplicação é somente um software? O hardware e software formam uma unidade? Existirá apenas

uma pessoa utilizando-o ou uma organização inteira [26]? O que o sistema abrangerá, ele automatizará todo o processo de negócio ou fará parte de um conjunto integrado de sistemas?

2.7.2 Identificando Atores

Identifique quais as pessoas e/ou sistemas que irão interagir com o sistema sendo construído, classificando-as segundo os papéis que as mesmas desempenham no sistema. Lembre-se que uma pessoa pode assumir mais de um papel dentro do mesmo sistema, gerando atores para cada papel ou atribuição encontrada.

Para cada ator, identifique seus objetivos com o sistema. Para uma análise do comportamento dos atores é interessante fazer as seguintes perguntas [39]:

- Quais são as tarefas de cada ator?
- O ator vai ler/alterar/escrever dados do sistema?
- Quais são as mudanças no ambiente que serão informadas ao sistema pelos atores?
- Quais as mudanças nos dados do sistema que deverão ser informadas aos atores?

2.7.3 Identificando UCs

Defina UCs que satisfaçam os objetivos dos atores. Nomeie-os de acordo com a natureza dos objetivos.

1. Escreva inicialmente o UC mais simples, ou seja, o principal cenário de sucesso.
2. Escreva os cenários alternativos a partir de alguma condição existente no cenário principal.
3. Escreva os cenários alternativos de insucesso (condições de falha). Os cenários complexos e grandes variações do UC em questão devem ser descritos em novos UCs. Utilize extensões de UC, se possível.
4. Examine os UCs para observar o potencial de reaproveitamento de partes comuns. A partir desta análise gere novos UCs que tenham grande probabilidade de serem reaproveitados através de relacionamentos de "include" em outras partes do sistema.

2.8 A Influência da Granularidade

O analista, que inicia o desenvolvimento dos UCs de um sistema, freqüentemente se depara com a pergunta: "Como eu devo separar os casos de uso?", ou então, "Devo criar um novo caso de uso, ou continuar a escrever neste mesmo?", "Devo detalhar esse caso um pouco mais?".

Não ter uma orientação sobre como particionar os UCs pode fazer com que a organização tenha uma documentação tão díspar que dificultaria a aplicação de técnicas de mensuração e predição de esforço, ao mesmo tempo que provoca dificuldades na compreensão e assimilação dos UCs.

De maneira geral, não existe uma regra para particionar os UCs. No entanto, Larman [26] apresenta uma sugestão para a divisão dos mesmos. Segundo ele, cada UC deveria representar uma "tarefa de usuário", ou então, sob uma definição mais formal, um "Processo Elementar de Negócio" (Elementary Business Process - EBP).

Larman [26] define EBP assim:

> *"Uma tarefa executada por uma pessoa em algum lugar em algum momento, em resposta a um evento de negócio, que agrega valor e deixa os dados em um estado consistente."*

Esta sugestão pode levar a uma abordagem para a construção dos UCs, nesse caso o analista de UC deve:

1. Encontrar os objetivos/metas dos usuários.
2. Definir um UC para cada objetivo a ser atendido.

A diretriz EBP também não deve ser encarada como um mandamento inquebrável, existem situações onde é desejável que a mesma não seja aplicada, por exemplo quando se busca identificar porções dos UCs que podem ser reutilizadas através do relacionamento de inclusão. Nessa atividade podem ser gerados UCs que não respeitam a diretriz, mas que são importantes para a construção de um modelo compreensível.

A diretriz deve, portanto, ser aplicada na determinação do nível adequado para a maioria dos UCs do sistema, não de todos eles.

2.9 Estilos de UCs

Existem diversos estilos para o desenvolvimento de UCs, existem estilos simples e outros bastante sofisticados. Os UCs podem diferir bastante com

relação a muitos aspectos. Cockburn [9] menciona ter encontrado 18 diferentes definições para UC, que diferem ao longo de 4 dimensões: propósito, conteúdo, pluralidade e estrutura. A combinação de algumas dessas dimensões forma um estilo de UC. A seguir, são abordados alguns aspectos relevantes dentro dos estilos mais conhecidos:

1. Escrita: Os UCs podem diferir com relação à forma de escrita de seu conteúdo. existem basicamente dois estilos:

   - Estilo de Escrita Essencial: Este estilo remete aos princípios da análise essencial [36], cuja premissa é identificar os objetivos do usuário sem incluir detalhes de implementação e/ou de interface. O importante é focar no propósito dos requisitos sem incorrer no erro de detalhar como os requisitos serão atendidos pelo sistema. Ex: O UC "efetuando saque de dinheiro" apresentado no ínicio do texto é um exemplo deste estilo de escrita.
   - Estilo Concreto: Decisões de interface e de forma de atendimento aos requisitos estão presentes neste estilo. Contrariamente ao estilo essencial, o UC pode, por exemplo, incluir detalhes sobre: formato de apresentação de informações, como caixas de diálogo; mecanismos de navegação pelo sistema, como o uso de botões;

     O texto abaixo apresenta o UC "Efetuando Saque de Dinheiro" reescrito segundo o estilo concreto.

     > *Efetuando saque de dinheiro: O cliente do banco chega até o caixa automático desejando sacar uma determinada quantia. O sistema apresenta uma caixa de diálogo onde o usuário pode digitar o número de sua conta corrente, após pressionar <ENTER>, o sistema exibe então uma caixa de diálogo para o preenchimento da senha, após pressionar <ENTER> novamente o sistema faz a validação e autenticação das informações e fornece um menu com opções ao cliente. As opções disponíveis são: 1. Consultar saldo, 2. Efetuar saque, 3. Sair. O cliente então solicita a operação de saque pressionando a tecla nº 2. O sistema exibe uma caixa de texto para que o cliente informe o valor a sacar. Após o cliente informar o valor e pressionar <ENTER>, o sistema entrega o dinheiro ao cliente, registrando o valor do saque efetuado e abatendo este valor do saldo do cliente.*

Note no exemplo anterior que o texto inclui diversas decisões de interface. Também pode ser observado o mecanismo de navegação, presente através da escolha de opções identificadas por teclas e botões.

2. Estruturação: Além do estilo de escrita, os UCs podem variar com relação à forma e subseções contempladas, os UCs mais completos freqüentemente incluem as seguintes seções:

   - Pré-condições: Indicam o estado que deve sempre ser verdadeiro para que o UC seja iniciado. As pré-condições não são testadas pelo UC, pelo contrário, são condições que são assumidas como sendo sempre verdadeiras. Tipicamente, uma pré-condição implica um cenário de outro UC que deve ter sido completado com sucesso [26] para que este UC se inicie.

     Ex: "O cliente deve ter validado seu cartão e senha no caixa automático", pode ser uma pré-condição para o UC "Efetuando Saque de Dinheiro".

   - Pós-condições: Indicam o estado do sistema que deverá vigorar após a execução do UC.

     Ex: "O saldo da conta do cliente foi atualizado e o cartão foi devolvido", pode ser uma pós-condição do UC "Efetuando Saque de Dinheiro".

   - Curso Básico ou Cenário Principal: Nesta seção fica descrita a principal interação entre o ator e o sistema na execução de uma determinada tarefa. Comumente descrito por um conjunto seqüencial de passos, o curso básico apresenta uma visão geral do UC.

   - Cursos ou Cenários Alternativos: Alguns escritores preferem descrever nesta seção os cenários alternativos ao cenário principal.

   - Exceções: Nesta seção tendem a ficar os cenários de insucesso, aqueles cenários que não levaram o ator a cumprir seus objetivos.

Observe na Figura 2.3, um exemplo do caso de uso "Efetuando Saque de Dinheiro"utilizando as seções que acabaram de ser descritas, entre outras mais.

Caso de Uso: Efetuando saque de dinheiro

Atores: Cliente do banco

Pré-Condições: 1. O cliente deve estar cadastrado no banco; 2. O cliente deve ter saldo disponível; 3. O caixa automático deve estar ligado e aguardando operação.

Cenário Principal:
1. O cliente chega até o caixa automático desejando sacar uma quantia em dinheiro.
2. O sistema do caixa automático está aguardando operação, emitindo uma mensagem que solicita ao cliente que insira seu cartão do banco.
3. O cliente insere o cartão do banco, o sistema valida o cartão e solicita a senha.
4. O cliente informa a sua senha.
5. O sistema valida a senha e fornece um menu com as seguintes opções: Consultar saldo, Efetuar saque, Sair.
6. O cliente seleciona Efetuar saque.
7. O sistema permite ao usuário a escolha do valor a sacar.
8. O cliente informa o valor a sacar e confirma a operação.
9. O sistema verifica o saldo, efetua o saque, entrega o dinheiro e diminui o saldo do cliente conforme o valor sacado.

Cursos Alternativos:
6A. O cliente escolhe Consultar saldo
6A1. O sistema emite um comprovante informando o saldo do cliente
6B. O cliente escolhe Sair
6B1. O sistema encerra a operação e devolve o cartão ao cliente.

Exceções
3A. O cartão inserido não pertence ao banco ou está com problemas.
3A1. O sistema emite mensagem informando o cliente sobre o problema ocorrido e volta a aguardar operação.
5A. A senha informada é inválida
5A1. O sistema emite mensagem informando o cliente sobre o problema ocorrido, solicita a senha novamente,até um máximo de três vezes.
9A1. O saldo é insuficiente para permitir o saque solicitado pelo usuário.
9A2. O sistema informa ao cliente que o saldo (valor do saldo) é insuficiente para completar o saque e volta a solicitar a operação desejada, passo 5.

Pós Condições: 1. O sistema retornou ao estado aguardando operação.

Figura 2.3: Um Caso de Uso Completo

2.10 Considerações Finais

Os UCs firmaram-se como o principal modelo de especificação de requisitos no desenvolvimento de software, tendo como suas principais vantagens: a simplicidade e a integração com os objetivos do usuário final da aplicação.

Não existe formalismo por trás da definição original dos UCs, razão pela qual existe uma disseminação de técnicas e mecanismos de composição dos mesmos. Embora isto cause uma certa confusão, isso pode ser visto como um ponto positivo, pois cada organização sente-se a vontade para definir o seu padrão de UCs da forma que lhe é mais conveniente.

Como afirmado por Larman [26], a preocupação excessiva com a construção de diagramas de UCs pode ser indício de inexperiência no uso da técnica, pois o fundamental em um caso de uso é o seu conteúdo e o detalhamento das funcionalidades, enfim, o principal é escrever os casos de uso. Por outro lado, os diagramas são necessários para permitir uma visão geral do sistema e para um entendimento adequado do relacionamento entre os diversos UCs. O tempo dispendido com cada artefato deve ser utilizado com parcimônia, sempre objetivando um melhor entendimento do sistema sendo especificado.

Outra consideração importante de Larman que deve ser exaustivamente reafirmada, é que os UCs não se restringem à tecnologia de Orientação a Objetos, pelo contrário, a técnica pode ser aplicada com sucesso em qualquer projeto de software.

No capítulo seguinte serão apresentados conceitos e definições da teoria dos conjuntos *Fuzzy*, muitos dos quais necessários para o entendimento dos capítulos subseqüentes.

Capítulo 3

Teoria dos Conjuntos Fuzzy

A sociedade atual, chamada de "sociedade da informação" e a maior parte do setor econômico dedicam-se à manipulação, processamento, seleção, armazenamento, disseminação, proteção, coleta, análise e classificação de informação, nossa melhor ferramenta para isto, é certamente, o computador [23].

Nas últimas décadas as tecnologias de banco de dados e sistemas operacionais evoluiram bastante, permitindo um manuseio mais veloz de uma quantidade de informações cada vez maior.

Ferramentas que auxiliam a análise e o entendimento dessas informações foram desenvolvidas, como por exemplo pode-se citar as tecnologias de Mineração de Dados (Data Mining) e os Sistemas de Suporte à Decisão (Decision Support Systems).

Apesar da abundância de informação, normalmente as análises têm que conviver com a ausência de informação ou com determinados graus de incerteza, o que agrega complexidade ao processo.

A influência que a incerteza exerce sobre a complexidade é bem descrito pelo exemplo de se dirigir um carro, extraído de [23].

"Pode-se concordar que a tarefa de dirigir um carro é (pelo menos relativamente) complexa. Além disso, dirigir um carro com transmissão manual é mais complexo que dirigir um carro com transmissão automática, um indicativo disso é que se faz necessário mais conhecimento para descrever como dirigir um carro no primeiro caso do que no último (por exemplo, é necessário conhecimento sobre rotações por minuto e como usar a embreagem). Entretanto a tarefa de dirigir um carro também envolve um grau de incerteza; por exemplo, nós não sabemos precisamente quando devemos parar ou desviar para evitar um obstáculo. Conforme o grau de incerteza aumenta - por exemplo, em tráfego pesado ou em estradas não familiares - aumenta a complexidade da tarefa. Portanto, percebe-se que a percepção de comple-

xidade aumenta tanto quando percebemos o quanto nós sabemos e quando percebemos o quanto não sabemos".

A incerteza e imprecisão fazem parte da linguagem natural e das formas tradicionais de se representar conhecimento e, portanto, não devem ser ignoradas. A imprecisão não significa perda de eficiência ou significado, por exemplo dizer que o dia está *chuvoso* pode ser mais útil e simples do que informar a quantidade de ml/hora que está chovendo, ou então, você pode simplesmente deduzir que seria demorado subir uma montanha porque percebe que ela é *alta*, sem no entanto saber sua altura exatamente.

Para uma expressão como *alta* ter significado ela não pode ser específica como por exemplo, 1000 metros, entretanto, ela também não pode ser arbitrária. Como classificar então uma montanha como *alta*? Montanhas de até 500 metros poderiam ser classificadas como baixas e montanhas a partir de 501 metros poderiam ser consideradas altas. Porém, não podemos aceitar que apenas um metro possa ser a diferença entre uma montanha ser baixa ou alta, é necessária uma transição gradual.

A teoria tradicional dos conjuntos divide os elementos do universo em dois grupos, o grupo dos membros e o grupo dos não membros de um dado conjunto. Um exemplo de conjunto dessa natureza pode ser o conjunto dos professores, ou uma pessoa é professor(a), ou não é.

No entanto, nem todos os conjuntos existentes podem ser tratados dessa forma, por exemplo o conjunto das pessoas com o cabelo de uma determinada cor ou das pessoas com obesidade. Conjuntos desse tipo não possuem limites definidos e a transição de um grupo para outro é gradual, não discreta.

Os conjuntos *Fuzzy* estendem a teoria tradicional de conjuntos permitindo a introdução de incerteza sobre os conjuntos, e possibilitando a mudança gradual através do conceito do grau de pertinência.

Um conjunto *Fuzzy* é caracterizado por uma função de pertinência [23], a função permite definir o quanto um indivíduo pertence a um determinado conjunto. Dessa forma, indíviduos podem pertencer em maior ou menor grau, podem ainda pertencer a mais de um grupo.

Ex: Uma pessoa que pesa 80 kg poderia ter um grau de pertinência de 40% ao grupo dos indivíduos de peso normal e de 50% ao grupo dos indivíduos obesos.

A teoria *Fuzzy* já é aplicada com sucesso em um variedade de segmentos que fazem tratamento de informação como engenharia, psicologia, inteligência artificial, medicina, sociologia e metereologia [23].

3.1 Conjuntos *Fuzzy*

A teoria clássica dos conjuntos define os membros e não membros de um dado conjunto através de uma função que retorna o valor 0 ou 1 para os elementos do conjunto universo, este tipo de conjunto é também chamado de *conjunto nítido*. Um conjunto *Fuzzy* é uma generalização desse conceito clássico, pois tal função poderia gerar valores pertencentes a um determinado intervalo, sendo que este valor indicaria o quanto o indivíduo pertence àquele conjunto.

O conjunto dos elementos definidos por tal função é denominado Conjunto *Fuzzy* [23].

3.2 Função de Pertinência

A função que determina o grau de pertinência de um indíviduo a um conjunto *Fuzzy* qualquer é denominada Função de Pertinência e é usualmente definida da forma:

$$\mu_A : X \rightarrow [0, 1]$$

X :	Conjunto Universo
μ_A :	Função de Pertinência
A :	Conjunto *Fuzzy*
[0,1] :	Intervalo fechado dos números reais

A viabilidade de utilização de um conjunto *Fuzzy* para modelar um conceito ou variável lingüística, depende do quão apropriada é a sua função de pertinência. Portanto, no processo de modelagem a maior preocupação certamente será a determinação prática de uma função precisa e justificável em qualquer situação particular. A maioria dos métodos propostos para se determinar uma função de pertinência tem fundamentação altamente empírica, e comumente envolvem a realização de experimentos sobre uma população de teste para medir a percepção de graus de pertinência para algum conceito [23].

Dentre as possibilidades encontradas na literatura, destacam-se duas famílias de funções [21, 35]:

- Funções Triangulares

$$\mu_{\overline{A}}(x) = \begin{cases} 0, & \text{se} \quad x < a \\ x - a, & \text{se} \quad x \in [a, m] \\ b - x, & \text{se} \quad x \in [m, b] \\ 0, & \text{se} \quad x > b, \end{cases}$$

- Funções Trapezoidais

$$\mu_{\overline{A}}(x) = \begin{cases} 0, & \text{se} \quad x < a \\ \frac{x-a}{m-a}, & \text{se} \quad x \in [a, m] \\ 1, & \text{se} \quad x[m, n] \\ \frac{b-x}{b-n}, & \text{se} \quad x \in [n, b] \\ 0, & \text{se} \quad x > b \end{cases}$$

As funções triangulares se mostram adequadas para trabalhar com valores de pertinência em torno de um ponto específico, já as funções trapezoidais são mais adequadas para tratar intervalos.

Veja na Figura 3.1 os dois exemplos de funções de pertinência [21].

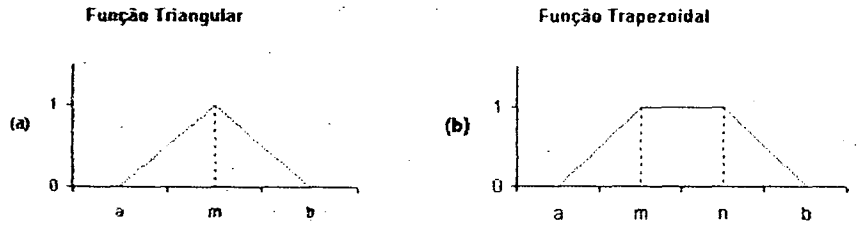

Figura 3.1: Funções de Pertinência

3.3 Definições Básicas

Para ilustrar melhor as definições sobre *Fuzzy*, considere a Tabela 3.1 que contém os valores de pertinência de um conjunto universo de pesos de indivíduos:

O conjunto universo X tem a seguinte definição:

Tabela 3.1: Conjunto Universo

Elementos (peso Kg)	Magro	Normal	Obeso	Muito Obeso
45	1	0.1	0	0
50	0.9	0.3	0	0
55	0.8	0.5	0	0
65	0.5	1	0	0
75	0	1	0	0
85	0	0.5	0.2	0
95	0	0	0.9	0.2
105	0	0	1	0.5
120	0	0	1	0.9

$$X = \{45,50,55,65,75,85,95,105,120\}.$$

- Suporte

  O *suporte* de um conjunto *Fuzzy* A no conjunto universo X é o conjunto que contém todos os elementos de X que tem valor de pertinência não nulo em A, o conjunto é dado pela Equação 3.1.

$$supp : P(X) \rightarrow P(x) \tag{3.1}$$

  onde,

$$\text{supp } A = \{x \in X / \mu_A(x) > 0\} \tag{3.2}$$

  Ex:

$$\text{supp(obeso)} = \{85,95,105,120\}$$

- Conjunto *Fuzzy* Vazio

  Um conjunto *Fuzzy* é dito vazio quando o seu *suporte* é vazio, isto é, todos os elementos possuem pertinência 0 em tal conjunto.

- Altura e/ou Supremo

  A *altura* de um conjunto *Fuzzy* é o maior grau de pertinência obtido por qualquer elemento dentro daquele conjunto.

  Ex:

$$\text{Altura(magro)} = 1$$

- Conjuntos normalizados

  Um conjunto *Fuzzy* é considerado normalizado se pelo menos um de seus elementos possui o mais alto grau de pertinência possível.

  No exemplo anterior, os valores de pertinência ficavam dentro do invervalo fechado [0,1], logo o mais alto valor possível é 1. Sendo assim, os conjuntos *Magro*, *Normal* e *Obeso* podem ser considerados como normalizados.

- Medidas *Fuzzy*

  Dado um elemento particular de um conjunto universo cuja pertinência nos vários subconjuntos deste conjunto ainda não é conhecida com certeza, uma medida *Fuzzy* associa um valor para cada um desses subconjuntos, que indica o grau de evidência de que tal elemento pertence ao subconjunto. Portanto, a medida *Fuzzy* é definida pela função:

  $$g : P(X) \rightarrow [0, 1]$$

  Veja o seguinte exemplo extraído de [23] que ilustra claramente as diferenças existentes entre a utilização de uma Medida *Fuzzy* e um Conjunto *Fuzzy*:

  Suponha uma classificação que separe as pessoas pela faixa etária da seguinte forma: Pessoas dos 0 aos 9 anos, dos 10 aos 19, dos 20 aos 29 e assim por diante... Tais conjuntos são nítidos, ou seja, os limites não são contínuos. Dado uma pessoa qualquer, pode-se tentar inferir com um determinado grau de certeza que a pessoa pertença a alguns dos grupos existentes. Este seria um exemplo de utilização de uma medida *Fuzzy*, determinando o grau de evidência de que uma pessoa pertence a alguma das faixas existentes.

  Ao contrário, se formularmos o mesmo problema em termos de conjuntos *Fuzzy*, em vez de determinarmos a que faixa etária pertence a idade daquela pessoa, teríamos que determinar a que grupo a pessoa pertence, se ao grupo dos jovens, adultos, idosos, etc. Sendo que estes conjuntos não são nítidos, seus limites não são precisos nem conhecidos.

3.4 Números *Fuzzy*

Klir e Folger [23] definem um número *Fuzzy* como "Um conjunto *Fuzzy* convexo e normalizado definido sobre o conjunto dos números reais R, cuja função de pertinência é contínua sobre tal conjunto".

Os números *Fuzzy* são geralmente aceitos como conjuntos *Fuzzy* triangulares, onde o pico corresponde ao valor dado [32]. Além da forma triangular, a trapezoidal é também bastante utilizada para representar os números *Fuzzy*, em casos especiais pode-se ainda optar por formas não simétricas [21].

3.5 Fuzificação

Quando se deseja modelar alguma idéia ou conceito discreto em termos de um conjunto *Fuzzy*, é necessário um processo de fuzificação.

Segundo Lima Júnior [21], "a fuzificação ocorre quando um conjunto *Fuzzy* $\tilde{A}$ é obtido pelo "alargamento" *Fuzzy* de um conjunto nítido, isto é, um conjunto nítido é convertido em um conjunto *Fuzzy* apropriado para expressar medidas de incerteza".

3.6 Defuzificação

Ao contrário do processo de fuzificação, o processo de defuzificar consiste em transformar as incertezas presentes em um conjunto *Fuzzy* em um número real e discreto, permitindo assim a sua manipulação em processos convencionais.

Existem diversos métodos clássicos de defuzificação, tais como [21, 27]:

- Método do Centro de Gravidade
- Média dos Máximos
- Média dos Pontos de Suporte

Embora tais métodos clássicos sejam comumente utilizados, Lima Júnior utiliza em [21] um critério próprio de defuzificação, pois segundo Leekwijck e Kerre [45], "critérios próprios de defuzificação, que não estão diretamente relacionados com conceitos e fundamentos teóricos, podem ser formulados quando houver resultados práticos de maior importância, considerando fatores como eficiência computacional e clareza do operador de defuzificação".

3.7 Operações sobre Conjuntos *Fuzzy*

Conforme proposto por Zadeh [49] e re-apresentado por Lima Júnior [21] várias das operações clássicas existentes para os conjuntos nítidos foram estendidas e adaptadas para a teoria *Fuzzy*, eis algumas delas:

- Igualdade
- Complemento
- Intersecção
- União
- Operações algébricas
  - Produto cartesiano
  - Soma algébrica
  - Soma de contorno
  - Diferença de contorno
  - Produto algébrico
  - Combinação convexa
- Agregação de conjuntos

Dentre estas operações, a que tem principal relevância neste trabalho é o *complemento Fuzzy*, e que está detalhado a seguir.

3.7.1 Complemento *Fuzzy* [23]

O complemento de um conjunto *Fuzzy* A é definido por uma função

$$c_A : [0,1] \rightarrow [0,1], \tag{3.3}$$

que associa um valor $c(\mu_A(x))$ para cada grau de pertinência $\mu_A(x)$. O valor associado é interpretado como o grau de pertinência do elemento x no conjunto *Fuzzy* que representa a negação do conceito representado por A.

Por exemplo, se A fosse o conjunto das pessoas com obesidade, seu complemento seria o conjunto das pessoas não obesas.

Existe a possibilidade de que diversos elementos tenham grau de pertinência não nulo em ambos os conjuntos [23].

Klir e Folger [23] sugerem que, quando os limites dos graus de pertinência estão restritos ao intervalo [0,1], o operador de complemento funciona exatamente da mesma maneira que o seu correspondente na teoria dos conjuntos nítidos tradicionais, ou seja, poderia ser obtido pela Fórmula 3.4.

$$c(\mu_A(x)) = 1 - \mu_A(x), \forall \, x \in X \tag{3.4}$$

3.8 Lógica *Fuzzy* [23]

A lógica proposicional clássica, também chamada de lógica booleana, é firmada sobre o princípio de que toda proposição assume um valor falso ou verdadeiro. Esse princípio da dualidade das proposições vem sendo questionado desde Aristóteles [23].

Proposições futuras não podem ser consideradas verdadeiras e nem tampouco falsas, pois potencialmente podem assumir qualquer um dos dois valores, sendo assim, o valor verdade dessa proposição é dito *indeterminado*.

No entanto, não apenas as proposições futuras sofrem do problema da indeterminação, por exemplo considere a seguinte afirmação:

"O céu está nublado!"

A expressão é verdadeira ou é falsa? Como caracterizar um céu nublado? Qual a quantidade de nuvens que diferencia um "céu nublado" de um "céu aberto"?

Sendo assim, a lógica booleana pode ser estendida em uma lógica na qual as proposições poderiam receber até 3 valores: verdadeiro, falso ou intederminado.

No entanto, fica claro que existe uma insuficiência das lógicas convencionais para trabalhar com as imprecisões do conhecimento humano e é justamente a capacidade de se obter um *raciocínio aproximado* a partir de proposições imprecisas o que caracteriza o "kernel"[1] da lógica *Fuzzy*, utilizando como principal ferramenta a teoria dos conjuntos *Fuzzy* [23].

O foco primário da lógica *Fuzzy* é na linguagem natural, onde o raciocínio aproximado com proposições imprecisas é bastante típico. O seguinte silogismo é um exemplo de raciocínio aproximado em termos linguísticos que não pode ser tratado pelos predicados lógicos clássicos.

Moedas velhas normalmente são coleções raras.
Coleções raras são caras.
Moedas velhas normalmente são caras.

Isto é uma inferência dedutiva significativa. Para trabalhar com inferências tais qual esta, a lógica *Fuzzy* permite o uso de vários termos que transformam a expressão em que estes se inserem:

- *predicados Fuzzy*: Caro, velho, raro, perigoso;
- *quantificadores Fuzzy*: Muitos, poucos, quase todos, usualmente;

[1] Núcleo

- *valores-verdade Fuzzy*: Razoavelmente verdadeiro, muito verdadeiro, mais ou menos verdadeiro, menos verdadeiro, principalmente falso;
- *modificadores Fuzzy*: Comumente, quase impossível, extremamente improvável.

Cada predicado simples como:

$$x \text{ é } P$$

é representado em lógica *Fuzzy* por um conjunto *Fuzzy*, conforme descrito previamente. Assuma, por exemplo, que x seja a idade de uma pessoa e que P seja o significado *jovem*. Então, assumindo que o conjunto universo é o conjunto dos números inteiros de 0 a 60 representando idades diferentes, o predicado pode ser representado por um conjunto *Fuzzy* cuja função de pertinência define o quanto cada idade pertence ao termo lingüístico *jovem*.

$$\mu_{Jovem} : [0, 1, ..60] \rightarrow [Jovem] \tag{3.5}$$

Considere agora que o valor verdade da proposição é obtida por uma substituição particular de x no predicado, tal como:

Tina é jovem.

O valor verdade desta proposição não depende apenas do grau de pertinência da idade de Tina no conjunto *Fuzzy* para caracterizar o conceito de uma pessoa jovem, mas também depende do quanto de veracidade (ou falsidade) foi requisitado. Veja alguns exemplos possíveis de quantificadores de veracidade:

- "Tina é jovem" é verdade.
- "Tina é jovem" é falso.
- "Tina é jovem" é razoavelmente verdade.
- "Tina é jovem" é muitíssimo falso.

A lógica *Fuzzy* tem sua operacionalidade baseada em diversas operações que utilizam conjuntos *Fuzzy*, possibilitando sua utilização em áreas do conhecimento humano em que o raciocínio é impreciso, tal qual a linguagem natural.

3.9 Considerações Finais

Conforme Klir e Folger afirmam em [23], a teoria *Fuzzy* é aplicada com sucesso em numerosos e diversos campos da ciência, tais como engenharia, psicologia, inteligência artificial, medicina, ecologia, reconhecimento de padrões, recuperação de informações, sociologia e metereologia.

O trabalho de Lima Júnior [21], também descrito no próximo capítulo, apresenta possibilidades da aplicação da teoria *Fuzzy* na extensão e adaptação de métricas de software convencionais, devido à constante carência de precisão das informações na atividade de planejamento e medição.

A aplicação da teoria se mostra particularmente apropriada para a manipulação e tratamento de informação juntamente com incerteza e imprecisão, o que certamente ocorre no caso da atividade de planejamento e medição de software.

Capítulo 4

Pontos por Função e Pontos por Caso de Uso

"Nos primórdios da computação, os custos com software comprometiam uma pequena parte dos orçamentos de um sistema de computação, devido ao alto custo do hardware utilizado. Atualmente, o software é o elemento mais caro da maioria dos sistemas de computação. Em sistemas complexos, um erro na estimativa de custo pode ser desastroso e fazer a diferença entre lucro e prejuízo" [36].

O contexto atual demanda maneiras eficientes de se estimar custo e esforço para a construção de software, mas para ser capaz de gerar tais estimativas, antes de tudo é necessário conseguir medi-lo adequadamente.

A primeira e mais conhecida métrica de software desenvolvida é a métrica de Linhas de Código (Lines of Code-LOC), que é uma medida do tamanho físico de um software e, possui em sua simplicidade, sua maior virtude e também o seu maior defeito. A simplicidade torna a LOC fácil de coletar e verificar, também esta simplicidade torna a LOC tecnologicamente dependente, ou seja, não é trivial efetuar comparações de LOC quando são utilizadas linguagens de programação diferentes.

Como alternativa às medidas físicas de tamanho, foram desenvolvidas as medidas funcionais. As métricas orientadas por função medem a funcionalidade presente na aplicação. As métricas funcionais foram introduzidas por Albrecht [2] com a Análise de Pontos por Função.

4.1 Análise de Pontos por Função

A Análise de Pontos por Função (Function Point Analysis-FPA) popularizou-se com a criação em 1986 do Grupo Internacional de Usuários de Pontos por

Tabela 4.1: Contagem de UFP

Parâmetro de medida	Qtde. x Peso Simples	Qtde. x Peso Médios	Qtde. x Peso Complexos	Total
N° Entradas do Usuário	[ ] x 3 +	[ ] x 4 +	[ ] x 6	=T1
N° de Saídas do Usuário	[ ] x 4 +	[ ] x 5 +	[ ] x 7	=T2
N° de Consultas do Usuário	[ ] x 3 +	[ ] x 4 +	[ ] x 6	=T3
N° de Arquivos	[ ] x 7 +	[ ] x 10 +	[ ] x 15	=T4
N° de Interfaces Externas	[ ] x 5 +	[ ] x 7 +	[ ] x 10	=T5

Função (International Function Point Users Group - IFPUG), que é formado por usuários interessados em efetuar padronizações adicionais na contagem de FP.

O primeiro passo na FPA é a determinação dos Pontos por Função não ajustados (Unajusted Function Points - UFP). A contagem dos UFP consiste, basicamente, no cálculo da Fórmula 4.1, baseado no preenchimento da Tabela 4.1 [36].

- N° de Entradas do Usuário (External Input - EI): São processos de entrada de dados. Cada entrada de informação que provê uma informação diferente dos demais processos de entrada é contabilizada. Entradas devem ser distinguidas de consultas, que são contadas separadamente.

- N° de Saídas do Usuário (External Output - EO): Cada processo de saída que forneça informações sobre dados do sistema é contabilizado neste item. Relatórios, telas, mensagens de erro também são contabilizadas. Itens de dados individuais dentro de um relatório não são contados separadamente.

- N° de Consultas do Usuário (External Inquiry - EQ): Uma consulta é definida como uma entrada on-line, que resulta na geração de alguma resposta imediata do software sob a forma de uma saída on-line. Cada consulta distinta é contabilizada.

- N° de Arquivos Lógicos Internos(Internal Logical Files - ILFs): São agrupamentos lógicos de dados mantidos dentro da aplicação, durante ou após a sua execução. Arranjos de informações e outros arquivos devem ser contabilizados, podendo ou não fazer parte de uma base de dados maior.

- Nº de Interfaces Externas (External Interface File - EIF): Todas as interfaces em linguagem de máquina que são usadas para transmitir informações a outros sistemas, tais como arquivos de transferência de dados, fitas de "backup", etc.

Cada item pertencente a um dos cinco grupos deve ser classificado quanto à complexidade, determinar essa complexidade tem um caráter subjetivo, contudo, algumas empresas procuraram estabelecer critérios de forma a minimizar distorções na caracterização da complexidade dos itens encontrados.

O IFPUG, através de seu manual [19], orienta essa classificação através de matrizes que classificam cada um dos grupos. Os valores considerados pelas matrizes são: o número de registros lógicos (Record Element Types - RET), o número de itens de dados referenciados (Data Element Types - DET) e o número de arquivos referenciados (File Types Referenced - FTR).

Após o preenchimento desta tabela, deve ser realizado o cálculo do Fator de Ajuste de Valor (Value Adjustment Factor - VAF); o VAF é a avaliação de quatorze características gerais do sistema, baseadas nas respostas a serem dadas para as perguntas apresentadas na Tabela 4.2. Cada uma das questões é respondida segundo a sua influência com relação ao sistema, baseada na escala da Tabela 4.3.

Para finalizar o cálculo, utiliza-se a Fórmula 4.3, que consiste em multiplicar o número de UFP pelo fator de ajuste calculado.

$$UFP = \sum_{i=1}^{5} T_i \tag{4.1}$$

T_i - Total de pontos de cada parâmetro de medida obtido da Tabela 4.1

$$VAF = 0.65 + (\sum_{i=1}^{14} N_i * 0.01) \tag{4.2}$$

N_i - Avaliação de influência de cada fator existente na Tabela 4.2

$$FP = UFP * VAF \tag{4.3}$$

Tabela 4.2: Características do Sistema, adaptada de [36]

Pergunta	Nível de Influência
1. O sistema requer salvamento (backup) e recuperação (recovery)?	N01
2. Comunicações de dados são necessárias?	N02
3. Há funções de processamento distribuído?	N03
4. O desempenho é crítico?	N04
5. O sistema vai ser executado em um ambiente operacional existente,intensamente utilizado?	N05
6. O sistema requer entrada de dados on-line	N06
7. A entrada de dados on-line exige que a transação de entrada seja construída através de várias telas ou operações?	N07
8. Os arquivos são atualizados on-line?	N08
9. As entradas, saídas, arquivos ou consultas são complexas?	N09
10. O processamento interno é complexo?	N10
11. O código deve ser projetado para ser reutilizado?	N11
12. A conversão e a instalação estão incluídas no projeto?	N12
13. O sistema deverá estar projetado para instalações múltiplas em diferentes organizações?	N13
14. A aplicação deverá estar projetada para facilitar modificações e para facilidade de uso pelo usuário?	N14

Tabela 4.3: Escala de Influência

Graduação da Escala
0. Sem influência
1. Esporadicamente
2. Moderadamente
3. Médio
4. Significativo
5. Essencial

Tabela 4.4: Contagem de UFP para o Sistema de Automação de Loja

Parâmetro de medida	Qtde. x Peso Simples	Qtde. x Peso Médios	Qtde. x Peso Complexos	Total
Entradas do Usuário	[4] x 3 +	[10] x 4 +	[10] x 6	=112
N° de Saídas do Usuário	[2] x 4 +	[4] x 5 +	[10] x 7	=98
N° de Consultas do Usuário	[ ] x 3 +	[5] x 4 +	[ ] x 6	=20
N° de Arquivos	[ ] x 7 +	[6] x 10 +	[ ] x 15	=60
N° de Interfaces Externas	[ ] x 5 +	[ ] x 7 +	[1] x 10	=10

Um exemplo de como contar o número de FP é dado a seguir:

1. Um sistema de controle e automação de uma loja apresentou as seguintes contagens:

   No. de Entradas do Usuário: 4 - Simples, 10 - Médias, 10 - Complexas

   No. de Saídas do Usuário: 2 - Simples, 4 - Médias, 10 - Complexas

   No. de Consultas Externas: 5 - Médias

   No. de Arquivos: 6 - Médios

   No. de Interfaces Externas: 1 - Complexa

2. Preenchendo-se a Tabela 4.1 obtém-se a Tabela 4.4, com os valores para o sistema de automação da loja.

3. Conforme a Tabela 4.4, a contagem total de UFP para tal sistema foi 300 (112+98+20+60+10). Agora é necessário ajustar o valor encontrado de acordo as características gerais do sistema.

4. O sistema requer Processamento Distribuído (item 3) esporadicamente e Atualização On-Line (item 8) como um item essencial, todas as demais características não exercem influência sobre o software.

5. O cálculo final está descrito pelas Fórmulas 4.4 e 4.5.

$$VAF = 0.65 + (\sum_{i=1}^{14} N_i * 0.01) = 0.65 + ((1+5) * 0.01) = 0.71 \quad (4.4)$$

$$FP = UFP * VAF = 300 * 0.71 = 213 \quad (4.5)$$

De posse da contagem de FP dos projetos, as empresas podem estimar o esforço necessário para a construção do software, cobrar consistentemente pelo seu software, efetuar análises comparativas de desempenho e produtividade de projetos, além de outras finalidades.

Os FP fornecem uma medida padronizada e normalizada das aplicações. Sua aplicação é possível em qualquer projeto de software. A seguir, outros pontos favoráveis à utilização da FPA [1]:

- A FPA é mantida por uma organização internacional sem fins lucrativos, o IFPUG, desde 1986.
- A FPA é representada no Brasil pelo Grupo Brasileiro de Usuários de Pontos por Função (Brazilian Function Point Users Group - BFPUG), além de empresas especializadas.
- O IFPUG mantém um programa mundial de certificação de especialistas em FPA, trazendo credibilidade aos especialistas certificados e tranqüilidade às organizações que podem contar com tais profissionais.
- A norma ISO/IEC 20926 padronizou internacionalmente a técnica, trazendo uniformidade na sua aplicação.
- Já existe um grande acervo de dados históricos de projetos que foram mensurados com a FPA, provendo uma ótima base para a realização de estudos, comparações e estimativas.

Os FP estão presentes em um grande número de contratos e licitações no Brasil [1], a técnica deixou há bastante tempo de ser teoria, e passou a ser prática.

4.2 Limitações dos Pontos por Função

A despeito da larga base de usuários e empresas adeptos da FPA, pesquisadores e corporações continuaram a desenvolvê-la, inserindo modificações, propondo extensões e melhorias à técnica.

FPA apresentou significativos avanços sobre a técnica de LOC, possibilitando por exemplo, que duas aplicações desenvolvidas com tecnologias diferentes e por equipes diferentes possam ter suas produtividades comparadas.

O VAF, aplicado na técnica original para ajustar o valor não ajustado (UFP), não é unanimidade. O valor final obtido com os FP não expressa

apenas o tamanho do sistema, mas também expressa a complexidade no desenvolvimento deste software. Para o cálculo do VAF é necessário um certo grau de subjetividade, empregado pelo responsável pela contagem. Essa subjetividade pode levar a desvios e diferenças quando estas análises são realizadas por pessoas diferentes. Devido a isto é que existe uma certa tendência em se omitir o fator de ajuste e considerar apenas o valor bruto obtido (UFP) [46].

Observa-se ainda outros problemas na técnica original de FPA e na versão mantida pelo IFPUG:

1. Existência de apenas três classificações para os componentes (simples, médios e complexos): esta classificação parecia adequada quando a técnica foi criada, hoje no entanto, essa classificação é simplificada demais para permitir uma classificação adequada.

2. Identificação de Arquivos Lógicos Internos: a definição proposta por Albrecht e refinada pelo IFPUG é bastante abrangente e de difícil interpretação por exemplo nos modernos sistemas de bancos de dados relacionais, os SGBDs.

3. Problemas com o aumento do tamanho do sistema: Symons[42] identificou problemas na mensuração da técnica de FPA conforme o tamanho do sistema ultrapassa os 400 FP, segundo ele o número de FP de sistemas grandes é sub-estimado quando comparado ao número de FP de sistemas menores.

Insatisfeito com essas e outras limitações, Charles Symons publicou a técnica que será apresentada na próxima seção [42]. Essa variação tem predominância sobre a versão original no Reino Unido e em muitos países da Europa.

4.3 Análise de Pontos por Função MKII

O trabalho de Symons [42] claramente se propõe a ser um sucessor do trabalho de Albrecht, possuindo grandes "intersecções" com o trabalho original. A intenção do autor era a de superar com sua técnica, a maioria das limitações originadas a partir do trabalho de Albrecht.

Enquanto a FPA considera o sistema como um composto dos cinco tipos de componente (entradas e saídas externas, consultas externas, arquivos lógicos internos e interfaces externas), MKII FPA, "enxerga" o sistema como sendo um combinado de transações lógicas discretas. Estas transações podem

ser definidas como tarefas realizadas pelos usuários do sistema. A seguir, a definição de Symons para transação [42]:

"Uma combinação única de entrada, processo e saída acionada por um evento de interesse do usuário, ou uma necessidade de obter informação".

Exemplos de transações lógicas podem ser:

- Cadastrar uma venda
- Exibir uma listagem de usuários do sistema
- Calcular as parcelas de uma venda a crédito.

Para determinar o tamanho de uma transação T, utiliza-se a Fórmula 4.6.

$$T = W_i * \sum EED + W_e * \sum ER + W_o * \sum ESD \tag{4.6}$$

Onde:
EED - Elemento de Entrada de Dados
ER - Entidades referenciadas
ESD - Elemento de Saída de Dados
W_i, W_e, W_o - Pesos utilizados para EED, ER e ESD respectivamente

Para efeito de cálculo, Symons [42] define uma entidade assim: "Qualquer coisa (objeto, real ou abstrato) do mundo real sobre o qual o sistema provê informação".

A média dos pesos utilizados nas organizações é: Wi = 0.58 , We = 1.66 e Wo = 0.28 [44].

O número final de pontos não ajustados é dado pela Fórmula 4.7, que compreende um somatório do tamanho de todas as transações encontradas no sistema:

$$UFPs = \sum_{i=1}^{i=n} T_i \tag{4.7}$$

Ti-Transação do Sistema

De maneira similar à FPA, o valor não ajustado deve sofrer o ajuste através do produto por um fator de ajuste. O Ajuste de Complexidade Técnica (Technical Complexity Adjustment - TCA), é similar ao VAF da técnica FPA, as diferenças são:

1. Foram adicionadas cinco novas características às quatorze originais.

2. Foi adicionado um novo coeficiente C (que deve ser obtido através de calibramento), capaz de corrigir os pesos apurados para os itens que compõem o TCA, sendo assim, a Fórmula 4.8 representa como fica o cálculo.

$$TCA = 0.65 + C * \sum_{i=1}^{i=19} TD_i \tag{4.8}$$

Onde:
C - Coeficiente de ajuste (valor médio atual das organizações é 0.005 [44])
TD_i - Total Degrees of Influence - Somatório da influência das dezenove características do software sendo desenvolvido, medidas numa escala de 1 a 5 (cálculo análogo ao $\sum N_i$ do VAF, exibido na Fórmula 4.2)

O cálculo final dos MKII FP, o Cálculo de Pontos por Função Ajustado (Adjusted Function Point Calculation - AFPC), é dado pela Fórmula 4.9.

$$AFPC = UFPs * TCA \tag{4.9}$$

Notadamente, o método de Symons parece mais simples devido a centralizar os esforços em determinar as tarefas realizadas pelo usuário (transações) e na contagem do número de entidades envolvidas, parecendo mais adequado para a aplicação em sistemas industriais tradicionais.

4.4 Pontos por Caso de Uso

A especificação dos requisitos através da escrita de UCs já é realidade nos projetos de desenvolvimento de software. Sendo assim, é desejável a utilização de uma métrica de software que leve em conta essa realidade. Foi provavelmente esse desejo e essa necessidade que levaram Gustav Karner da empresa Objectory (atualmente Rational© Software [38]) a criar em 1993 a técnica de Pontos por Caso de Uso (Use Case Points - UCP) [22].

Os UCP são uma adaptação da FPA [2] e da técnica MKII FPA [44] para a utilização com UCs, a maioria das características da FPA está presente no UCP, como a contagem de pontos não ajustados e a aplicação de fatores de ajuste.

Assim que os primeiros UCs do sistema ficam prontos, a análise já pode ser iniciada. O formato e a granularidade dos UCs pode influenciar muito na

Tabela 4.5: Pesos dos Atores por Complexidade [16]

Tipo de Ator	Peso	Descrição-Exemplo
Ator simples	1	Outro sistema acessado através de uma API de programação
Ator médio	2	Outro sistema interagindo através de um protocolo de comunicação, como TCP/IP ou FTP
Ator complexo	3	Um usuário interagindo através de uma interface gráfica (stand- alone ou Web)

contagem dos UCP, o UC não pode ser demasiadamente abstrato ou de alto nível, nem tampouco pode ser de nível demasiadamente baixo. A discussão do nível adequado rende muitos artigos técnicos e científicos, mais informações sobre como decompor os UCs pode ser encontrada em [26].

4.4.1 Roteiro para Cálculo dos UCP [16]

Cada UC encontrado no sistema deverá ser avaliado e classificado. Todos os atores encontrados nos UCs também devem ser classificados. Com base em uma avaliação do peso de todos os atores, peso de todos os UCs e, nos fatores de ajuste, é que o número de UCP é calculado. Para o cálculo, deve ser utilizado o roteiro a seguir:

1. Classificação dos Atores do Sistema: todos os atores encontrados nos UCs do sistema devem ser classificados segundo a sua complexidade. A pontuação dos pesos dos atores é realizada utilizando-se a Tabela 4.5 .

2. Contagem do Peso dos Atores: O Peso não ajustado dos atores (Unajusted Actor Weight - UAW) é calculado somando-se os produtos do número de atores de cada tipo pelo seu respectivo peso, conforme a Fórmula 4.10.

$$UAW = \sum AtS * PAtS + \sum AtM * PAtM + \sum AtC * PAtC \quad (4.10)$$

Tabela 4.6: Pesos dos UCs por Número de Transações [16]

Tipo de UC	Número de transações	Peso
Simples	Até 3	1
Médio	4 a 7	2
Complexo	7 ou mais	3

AtS: Somatório do número de Atores Simples

PAtS: Peso de um Ator Simples, obtido da Tabela 4.5.

AtM: Somatório do número de Atores Médios

PAtM: Peso de um Ator Médio, obtido da Tabela 4.5.

AtC: Somatório do número de Atores Complexos

PAtC: Peso de um Ator Complexo, obtido da Tabela 4.5.

Ex: Um sistema possui:

- 3 atores: - 2 usuários, um gerente e um usuário comum (Atores Complexos)
- 1 sistema acessado via protocolo (Ator Médio). Neste caso o UAW fica como na Fórmula 4.11.

$$UAW = 2 * 3 + 1 * 2 + 0 * 1 = 8 \qquad (4.11)$$

3. Classificando os UCs: após o cálculo do UAW, todos os UCs serão avaliados. De maneira similar aos atores, os UCs são divididos segundo sua complexidade, conforme pode ser observado na Tabela 4.6.

   A classificação usa, primordialmente, o número de transações do UC. Uma transação é uma série de processos que devem ocorrer conjuntamente, ou então, serem cancelados em sua totalidade caso um dos processos falhar.

   A classificação dos UCs também pode ser realizada de outra maneira, através da identificação do número de entidades participantes do caso, após obter o número de entidades aplica-se a Tabela 4.7. Entende-se por entidade cada conceito de negócio (objeto do mundo real) encontrado no UC, um conceito pode ser uma idéia, uma coisa, ou um objeto. Ex: Dinheiro, Saque, Banco, Saldo.

Tabela 4.7: Peso de UCs por Número de Entidades [16]

Tipo de UC	Número de entidades	Peso
Simples	5 ou menos	1
Médio	5 a 10	2
Complexo	Mais de 10	3

4. Contagem do Peso dos UCs: O Peso dos Casos de Uso não Ajustado (Unajusted Use Case Weight - UUCW), é realizado da mesma forma que o cálculo do UAW, soma-se o produto do número de UCs de cada tipo pelo respectivo peso.

   Após a quantificação das transações ou entidades, os UCs devem ser classificados de acordo com as Tabelas 4.6 ou 4.7, conforme o caso e de maneira análoga às anteriores.

$$UUCW = \sum UCS * PUCS + \sum UCM * PUCM + \sum UCC * PUCC \quad (4.12)$$

UCS: Somatório do número de UCs Simples
PUCS: Peso de um UC Simples, obtido das Tabelas 4.6 ou 4.7.
UCM: Somatório do número de UCs Médios
PUCM: Peso de um UC Médio, obtido das Tabelas 4.6 ou 4.7.
UCC: Somatório do número de UCs Complexos
PUCC: Peso de um UC Complexo, obtido das Tabelas 4.6 ou 4.7.

O valor dos UUCP (não ajustado) é dado pela soma do peso dos atores (UAW) com o peso dos casos de uso (UUCW) (Fórmula 4.13).

$$UUCP = UAW + UUCW \quad (4.13)$$

Da mesma maneira que na técnica FPA, o valor de UCP deve ser ajustado por um multiplicador de fatores técnicos que envolvem o sistema. A novidade apresentada por essa técnica é a existência de um segundo multiplicador, o fator de ajuste ambiental.

O Fator de Ajuste Técnico (Technical Complexity Factor-TCF) engloba características que tendem a dificultar o processo de desenvolvimento, como

por exemplo a necessidade de se ter processamento distribuído, esse fator de ajuste tende a elevar o valor total de UCP.

Para calcular o TCF utiliza-se a Tabela 4.8. O cálculo do TCF é dado pela Fórmula 4.14.

$$TCF = 0.6 + (0.01 * \sum_{i=1}^{i=13} T_i) \tag{4.14}$$

Onde:
$T_i = I_i * P_i$
I_i - Nível de influência de um fator técnico
P_i - Peso associado ao fator técnico considerado

Tabela 4.8: Peso de Fatores Técnicos, adaptada de [16]

Fator	Requisito	Influência	Peso	Total
F1	Sistema distribuído	I1	2	T1
F2	Tempo de resposta	I2	2	T2
F3	Eficiência	I3	1	T3
F4	Processamento complexo	I4	1	T4
F5	Código reusável	I5	1	T5
F6	Facilidade de instalação	I6	0.5	T6
F7	Facilidade de uso	I7	0.5	T7
F8	Portabilidade	I8	2	T8
F9	Facilidade de mudança	I9	1	T9
F10	Concorrência	I10	1	T10
F11	Recursos de segurança	I11	1	T11
F12	Acessível por terceiros	I12	1	T12
F13	Requer treinamento especial	I13	1	T13

A presença dos fatores (Fi) é avaliada em uma escala de 0 (sem influência) a 5 (item imprescindível). A classificação (Ii) deve ser multiplicada pelo peso do item para chegar ao peso efetivo do fator, Ti.

O Fator de Ajuste Ambiental (Environmental Factor-EF), contrariamente ao TCF, engloba características desejáveis em um ambiente de desenvolvimento, portanto, a presença de tais características tende a diminuir a contagem total de UCP do sistema e, por conseqüência, o esforço dispendido na construção do software.

O cálculo do EF se dá através da avaliação da influência dos itens da Tabela 4.9. Novamente a influência dos itens é classificada em uma escala de 0 a 5, onde 0 indica a ausência de influência e 5 indica que determinado item está presente em todas as etapas do projeto. Ex: Uma avaliação de influência

5 do item E4 indica que a equipe dispõe de um analista experiente em tempo integral, o que contribui para a melhora dos índices de produtividade de um projeto.

Tabela 4.9: Pesos de Fatores Ambientais, adaptada de [16]

Fator	Descrição	Influência	Peso	Total
E1	Familiaridade com RUP ou outro processo formal	I1	1.5	T1
E2	Experiência com a aplicação em desenvolvimento	I2	0.5	T2
E3	Experiência em OO	I3	1	T3
E4	Presença de analista experiente	I4	0.5	T4
E5	Motivação	I5	1	T5
E6	Requisitos estáveis	I6	2	T6
E7	Desenvolvimento em meio-expediente	I7	-1	T7
E8	Linguagem de programação difícil	I8	2	T8

O cálculo do EF é dado pela Fórmula 4.15.

$$EF = 1.4 + (-0.03 * \sum_{i=1}^{i=8} T_i) \tag{4.15}$$

i, obtido da Tabela 4.9
Ti = Ii * Pi

Após o cálculo dos TCF e EF, a análise pode ser concluída perfazendo-se o cálculo final dos UCP através da Fórmula 4.16.

$$UCP = UUCP * TCF * EF \tag{4.16}$$

Com o cálculo dos UCP finalizado, é possível fazer estimativas do esforço necessário na construção do sistema. No trabalho original de Karner [22], sugere-se que para efeito de estimativas, um UCP equivale a 20 horas. Trabalhos subseqüentes sugerem que esse número pode variar de 15 a 30 horas em função do TCF e EF [3].

4.5 Trabalhos Relacionados

Muitos autores e pesquisadores buscaram soluções que superassem as limitações da técnica de Pontos por Função na medição de sistemas orientados

a objetos, outros buscaram criar soluções que automatizassem de alguma forma o processo de medição.

As técnicas que fazem a extração de métricas a partir de modelos da UML são de particular interesse neste trabalho, pois se relacionam com as métricas aqui propostas, que extraem medidas a partir do modelo de UCs, também pertencente à UML.

A seguir são apresentadas algumas destas experiências.

4.5.1 Pontos por Função Utilizando Modelos UML

Caldiera et al [7] desenvolveram uma ferramenta para a contagem de FP de sistemas OO. Neste trabalho, classes eram consideradas como arquivos lógicos e métodos como funções transacionais. Métodos abstratos não eram considerados e métodos concretos eram considerados apenas uma vez e na classe em que os mesmos eram declarados. Os métodos tinham seu peso definido de acordo com suas respectivas assinaturas.

Uemura et al apresentam em [43] uma ferramenta para a contagem de FP a partir de diagramas de classe e diagramas de seqüência. Os diagramas são obtidos automaticamente a partir de especificações geradas com a ferramenta CASE Rational Rose [18].

Os autores estabelecem regras para transformar alguns produtos dos diagramas (número de classes, o número de atributos de cada classe, mensagens trocadas por elas) nos artefatos contabilizados pela técnica FPA (Arquivos, Interfaces externas, etc).

Os resultados decorrentes da aplicação da ferramenta apresentados em [43] são bastante similares aos resultados obtidos por um especialista em FPA, que naquele trabalho para efeito de experimento, aplicou a técnica manualmente. Porém, essa comparação se restringia a um único e pequeno projeto, sendo necessário estender o experimento.

Kusumoto et al [24] deram continuidade ao trabalho iniciado em [43] construindo uma ferramenta que, nos mesmos moldes da ferramenta anteriormente concebida, calculava FP a partir de requerimentos expressos em uma ferramenta de extração e organização de requisitos da empresa Hitachi Ltda., o REQUARIO.

O experimento de Kusumoto et al[24] tem motivação semelhante à deste trabalho, pois a extração de métricas se dá a partir de documentos de requisitos. Conforme exposto no artigo, nota-se uma semelhança entre a especificação de requisitos utilizada na REQUARIO e um UC, conforme ilustrado a seguir:

Uma especificação de requisito REQUARIO é composta de:

- Character (Caracter) = Usuário do sistema
- Story (História) = Conjunto de exemplos de utilização da funcionalidade
- Scenario and Scene = Curso de exemplos dentro de uma Story.

Caso de Uso: O próprio UC é uma história de utilização do sistema.

- Ator = Usuário do Sistema
- Cenários = Cursos que podem ser seguidos pelo usuário durante a utilização da funcionalidade descrita no UC.

Marchesi apresenta em [28] um conjunto de métricas para também serem extraídas a partir de diagramas UML, no entanto, a proposta agora não é gerar um número de FP a partir das especificações, mas sim constituir uma nova métrica. O trabalho desenvolvido por Marchesi considera, basicamente, diagramas de classe e diagramas de casos de uso.

Diversas informações presentes no diagrama de classes são utilizadas na concepção das métricas, tais como: número de classes, número de responsabilidades de uma classe, hierarquia, entre outras mais.

Marchesi obtém a partir dos diagramas de UC, quatro métricas, enumeradas a seguir:

1. UC1 = Número de UCs
2. UC2 = Número de comunicações entre atores e UCs
3. UC3 = Número de comunicações entre atores e UCs, eliminando-se a redundância provocada pelos relacionamentos de uso e extensão.
4. UC4 = Uma estimativa global da complexidade do sistema, é uma fórmula baseada nos valores das três métricas anteriores.

Na mesma linha do trabalho de Marchesi, Ram e Raju [37] desenvolveram os Pontos por Função de Projeto Orientado a Objetos (Object Oriented Design Function Points - OODFP). Os OODFP são uma métrica similar à FPA, que também visa extrair dados a partir dos modelos obtidos na fase de projeto, para isso, Ram e Raju definiram como se medir a complexidade de uma classe e de seus métodos. A técnica de OODFP também obtém um valor não ajustado ao qual devem ser aplicados os mesmos fatores de ajuste da técnica tradicional.

Fetcke et al [15] ao contrário dos trabalhos anteriormente citados não propuseram novas métricas, mas sim objetivaram demonstrar a aplicabilidade da técnica FPA em sistemas orientados a objetos, para isso desenvolveram uma técnica para mapear em FP, artefatos do Processo de Desenvolvimento Orientado a Objetos (Object Oriented Sofware Engineering) criado por Jacobson et al [20].

No trabalho de Fetcke et al, os atores do sistema são utilizados para se identificar os limites do sistema a ser analisado segundo a técnica FPA, já para os casos de uso a técnica sugere um mapeamento em transações, ou seja, os casos de uso serão utilizados para se medir o número e a complexidade das transações do sistema. A técnica também utiliza informações da análise de objetos do domínio para completar o mapeamento.

O IFPUG [19] também apresenta sua visão sobre a contagem de FP em sistemas orientados a objeto, segundo o instituto, as classes deveriam ser consideradas como arquivos lógicos e os métodos como transações (de maneira similar à técnica de Caldiera), no entanto, essa abordagem não comporta características inerentes aos sistemas OO, como a herança e comunicação entre os objetos.

4.5.2 Aplicação de Lógica *Fuzzy* nos Fatores de Ajuste da FPA

Buscando diminuir os problemas causados pela subjetividade possivelmente empregada nos fatores de ajuste, alguns autores sugerem a aplicação de lógica Fuzzy no processo de ajuste de valor, encontrado na FPA.

Yau e Tsoi [48] sugerem que, devido à incerteza comumente encontrada no início dos projetos e nos processos de estimativa de custo, seria mais fácil para os responsáveis pela aplicação das métricas, determinar o grau de influência sobre as características do sistema (Tabela 4.2) através de descrições cognitivas ao invés de valores numéricos. Por exemplo, seria mais simples utilizar descrições como "Muito provavelmente 3", "Improvavelmente 2", "Extremamente improvável 1", que classificar a influência das características do sistema segundo um simples valor numérico.

Através da construção de funções *Fuzzy*, é possível a obtenção de intervalos a partir de uma avaliação realizada pelo usuário. Veja o seguinte exemplo transcrito de [48]:

> *Suponha que o analista responsável pela estimativa defina que o grau de complexidade da aplicação é "Provavelmente 2" e "Improvavelmente 1". A primeira expressão "Provavelmente 2" conduz a um intervalo de 1.5 a 2.5, e a segunda expressão, "Improvavelmente 1" leva a um intervalo que consiste de duas partes, de 0 a 0.25 e de 1.75 a 5. O intervalo resultante varia de 1.75 a 2.5.*

Através da combinação dos intervalos obtidos com cada uma das características do sistema, obtém-se um intervalo final para o número de FP da aplicação. Ex: O sistema terá de 230 a 245 FPs.

Dessa forma, a lógica Fuzzy transforma as impressões do analista em um intervalo de valores, resultando em maior confiabilidade e capacidade de predição, possibilitando ao gerente de projetos ter maior controle sobre as decisões a serem tomadas.

4.5.3 Análise de Pontos por Função Fuzzy

Lima Júnior em [21] cita que medidas derivadas a partir de pontos por função, como custo e prazo, podem estimar valores não factíveis em conseqüência do modo atual de classificar a funcionalidade das funções que constituem o sistema, em termos de complexidade.

Atualmente, a FPA divide a complexidade das funcionalidades em três classes, sendo que cada uma possui um determinado valor em FP. As classes são:

- Baixa
- Média
- Alta

A primeira limitação dessa classificação apontada por Lima Júnior [21] é quanto ao limite da classificação *Alta*, que se mostra insuficiente para diferenciar funcionalidades que ultrapassam este limite. Veja o seguinte exemplo:

- Função 1 - 1 DET e 50 RETs - Complexidade Alta
- Função 2 - 1 DET e 200 RETs - Complexidade Alta

No exemplo anterior, a FPA disporia em uma mesma categoria, funcionalidades que possuem tamanhos diferentes. Tal fato pode também comprometer as estimativas de projetos de manutenção. Observe o seguinte exemplo típico de uma manutenção evolutiva:

> Um EIF era constituído de 2 RETs e 55 DETs, ou seja, era classificado como de complexidade *alta*, medindo portanto 10 FP.
>
> Após uma manutenção que adicionou funcionalidade à aplicação, o EIF mencionado passou a ser constituído de 2 RET e 85 DETs, no entanto a FPA continua classificando-o como de complexidade alta e sendo equivalente a 10 FP.
>
> Resumindo, houve um acréscimo de 30 DETs que não é percebido pela FPA.

O problema não se resume apenas a uma limitação da faixa de complexidade alta, mas sim ao fato da classificação não ser contínua, mas sim discreta, pois sempre que houver um acréscimo de funcionalidade que não for capaz de mudar a faixa de complexidade em que a mesma está inserida (simples para média, média para complexa), este acréscimo não será percebido.

Lima Júnior divide seu trabalho então em 4 etapas que transformam e estendem a FPA na Análise de Pontos por Função Fuzzy (Fuzzy Function Points Analysis - FFPA) [21], são elas:

1. Fuzificação dos termos lingüísticos[1] das matrizes de complexidade da FPA, através da geração de números *Fuzzy* (trapezoidais);
2. Extensão das matrizes de complexidade da FPA, gerando-se novos termos lingüísticos;
3. Determinação do valor em pontos por função dos novos termos lingüísticos, gerados na etapa anterior; e
4. Defuzificação dos valores dos termos lingüísticos da FFPA em pontos por função.

O primeiro e o último passos são os principais para a extensão do modelo clássico em um modelo *Fuzzy*, tratando principalmente da construção de uma classificação contínua para a complexidade das funções a serem medidas.

[1] No caso, "termos lingüísticos" referem-se às descrições cognitivas utilizadas para denominar cada faixa de complexidade, como por exemplo os termos "Simples" ou "Complexo".

Fuzificação dos Termos Lingüísticos

Esta etapa transformou as matrizes de complexidade da FPA, cada linha das matrizes foi transformada em um número *Fuzzy* trapezoidal [23].

Basicamente, isso consiste em transformar a classificação que era discreta em um intervalo contínuo, alongando e criando intersecções entre as faixas de complexidade, veja o exemplo da primeira linha da matriz de complexidade de um ILF com 1 RET, a Tabela 4.10 foi transformada no número trapezoidal apresentado na Figura 4.1.

Tabela 4.10: Matriz de Complexidade de um ILF - 1a. linha

Número de Registros Lógicos - RETs	**Itens de Dados Referenciados - DETs**		
	1 a 19	**20 a 50**	**51 ou mais**
1	BAIXA	BAIXA	MÉDIA

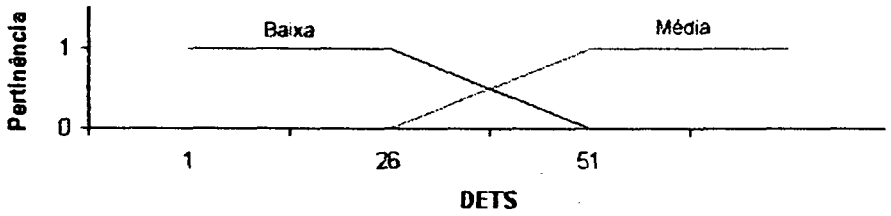

Figura 4.1: Números *Fuzzy* Trapezoidais Correspondentes a ILFs com 1 RET

Extensão das Matrizes de Complexidade e Determinação do seu Valor em Pontos por Função

O segundo e o terceiro passo tratam da criação de uma nova faixa de complexidade, *Muito Alta*, que possibilitaria a extensão das matrizes atuais da FPA. Segundo o autor, cada organização poderia ajustá-la de forma a melhor classificar suas funcionalidades.

De fato, uma faixa variável permite uma mensuração adequada, porém ao permitir isso a técnica impede a realização de comparações inter-organizações, pois cada uma pode ter a sua própria definição para tal faixa, assim 400 FFPA para uma organização podem não significar o mesmo que 400 FFPA para outra (mesmo que fossem considerados apenas os pontos não ajustados).

Defuzificação dos Valores dos Termos Lingüísticos

Esta etapa procura estabelecer critérios para que dada uma função que se deseja classificar e suas respectivas medidas, seja possível obter através do número *Fuzzy* trapezoidal, um número em pontos por função.

O próprio autor cita que não foram utilizados os métodos clássicos de defuzificação, sendo então criado um método próprio de defuzificação que está transcrito a seguir:

Em FFPA [21], para se obter o número de pontos por função p_d a partir dos números *Fuzzy* trapezoidais, há dois casos possíveis:

- O número de DETs está entre os valores de m e n do número *Fuzzy* trapezoidal. Nesta situação, o valor em FPA é igual ao valor dos pontos por função em FFPA, pois $\mu_{\overline{N}}(x) = 1$
- O número de DETs está entre os valores de n_i e b_i do número *Fuzzy* trapezoidal, implicando que o mesmo também está entre a_{i+1} e m_{i+1}. Neste caso, realiza-se o seguinte processo de defuzificação, onde $\overline{\mu}_{\overline{N}}(x)$ é o complementar de $\mu_{\overline{N}}(x)$.

  $p_d = \mu_{\overline{N}}(x).p_i + \overline{\mu}_{\overline{N}}(x).p_{i+1}$

Por exemplo, aplicando a definição acima para um ILF com 1 RET e 35 DETs, obtém-se:

$\mu_{\overline{N}}(35) = (51-35)/(51-26) = 0.64$; logo, o complementar $\overline{\mu}_{\overline{N}}(35) = 0.36$
$p_d = 0.64^*(7) + 0.36^*(10) = 8.08$ FP

Lima Júnior [21] discorrendo sobre a extensão da FPA afirma o seguinte: "o uso de termos lingüísticos permite a substituição dos intervalos clássicos por conjuntos *Fuzzy*, criando-se uma extensão do modelo original". Citam ainda algumas vantagens decorrentes da extensão:

- O modelo se torna mais genérico;
- O modelo simula a maneira como os seres humanos interpretam termos lingüísticos; e
- A transição de um termo lingüístico para um termo lingüístico contíguo se torna gradual, ao contrário da forma abrupta original.

A FFPA apresenta um avanço sobre a FPA na mensuração das funcionalidades, pois minimiza alguns dos principais problemas já presentes na técnica tradicional.

4.6 Ferramentas para Extração de Métricas

A aplicação manual das métricas é um trabalho bastante custoso, que envolve uma série de cálculos e anotações, dificultando e tornando propensa a erros a atividade de medição. Sob esse prisma, torna-se bastante interessante a utilização de ferramentas que automatizem e/ou orientem tal processo.

As ferramentas que trabalham com UCs e aplicação de métricas se dividem basicamente nos seguintes grupos:

- Ferramentas de especificação e organização de requisitos
- Ferramentas de modelagem
- Ferramentas de estimativa de custo e esforço
- Ferramentas para coleta de métricas específicas

As ferramentas de especificação e de modelagem não são objeto direto de estudo neste trabalho, eis alguns exemplos de ferramentas assim: Rational Requisite Pro [17], Borland® CaliberRM [4], Borland® Together [5] e IBM Rational Rose® [18].

4.6.1 Ferramentas de Estimativa de Custo e Esforço

As ferramentas que auxiliam o processo de predição de custo e esforço em sua grande maioria estão associadas ao modelo de predição COCOMO (Constructive Cost Model) [40].

Eis algumas ferramentas e suas características:

- Costar [11]: Ferramenta de estimativa de custo de software, baseada no COCOMO. Um gerente de projeto de software pode usar o COSTAR para efetuar diversas simulações com variáveis relacionadas ao projeto (tamanho e experiência da equipe, tecnologia empregada, etc), produzindo estimativas de duração, esforço e custo de um projeto.
- Cost Xpert [10]: É um software desenvolvido para estimar tempo e custo relacionado com o desenvolvimento de projetos de software. Para estimar o tamanho do projeto pode-se utilizar sete métodos para cálculo: SLOC, Function Points, Feature Points, GUI Metrics, Object Métrics, Bottom Up e Top Down.

4.6.2 Ferramentas para Coleta de Métricas Específicas

Existem algumas outras ferramentas destinadas à coleta de métricas específicas. Destacam-se neste segmento as ferramentas para aplicação da FPA.

Ferramentas para a FPA

A grande maioria das ferramentas de automatização da FPA permite que o lançamento dos valores de cada etapa da análise seja realizado na própria ferramenta [29], ficando esta responsável pela contabilização e organização das avaliações realizadas.

Tais ferramentas permitem que se armazenem registros históricos dos projetos mensurados, possibilitando a realização de avaliações e comparações entre os projetos.

Ferramentas para o UCP

Devido à popularidade da técnica de UCP ser bastante inferior à da técnica FPA, o número de ferramentas disponíveis para aplicação da técnica é também muito inferior. Alguns exemplos são o Estimate Easy Use Case [13] e o Enterprise Architect [12].

Também é notória a escassez de ferramentas que sejam destinadas exclusivamente à especificação de requisitos com UCs.

A ausência de ferramentas que orientem a especificação de UCs e que também permitam a aplicação automática da técnica UCP, propicia um meio favorável para o lançamento de novidades no setor.

4.7 Considerações Finais

Ambas as técnicas, FPA e UCP, possuem vantagens e desvantagens. A FPA é, naturalmente, mais madura, devido a anos de aprimoramento através das entidades mantenedoras. Já a técnica de UCP é bastante promissora devido a sua integração com a documentação gerada na fase de levantamento e especificação de requisitos, no entanto, a mesma carece ainda de uma formalização e padronização.

Para a aplicação da FPA com UC, o analista deve efetuar transposições de paradigmas, pois tal aplicação exige que o analista obtenha informações que não estão explícitas em um UC, como por exemplo: número de transações e número de consultas.

Particularmente importante para o sucesso da aplicação da técnica de UCP é a determinação da granularidade ideal para os UCs. O UCP carece

de uma organização ou entidade que efetue padronizações e a disseminação da métrica, da mesma forma que ocorreu com a FPA.

Os UCP parecem uma boa alternativa para as organizações, que estão iniciando na utilização de UC e na coleta de métricas. No entanto, a métrica ainda apresenta algumas limitações, como por exemplo as possibilidades limitadas de classificação do tamanho dos UCs.

Embora a FFPA apresente avanços significativos na construção de classificações graduais para as funcionalidades analisadas, ela sofre da mesma deficiência que a FPA quando se deseja medir UCs, pois as informações necessárias para sua aplicação não estão diretamente expressas em um UC e, talvez, tampouco estejam disponíveis nas etapas iniciais do desenvolvimento de um software.

O próximo capítulo apresenta duas novas métricas que, da mesma forma que o UCP, foram desenvolvidas especialmente para a medição de UCs. As métricas apresentam soluções para alguns dos problemas encontrados no UCP, possibilitando inclusive a obtenção de uma medida de tamanho para cada UC em separado.

Independentemente da métrica adotada é muito importante para as organizações a coleta efetiva das métricas e o registro histórico dos projetos, formando uma base de conhecimento que auxilie no processo de tomada de decisão.

Capítulo 5

Métricas Baseadas em Casos de Uso e Teoria *Fuzzy*

Analisando a técnica UCP descrita no Capítulo 4 podem ser observadas diversas limitações na mesma, algumas das quais podem provocar dificuldades para a obtenção efetiva do tamanho da funcionalidade expressa no UC.

Este capítulo apresenta uma nova proposta de métrica para UCs; esta métrica extrai informações da estrutura e do conteúdo de um UC para determinar seu tamanho.

Antes de explorar as características da nova métrica é conveniente entender quais são os principais problemas da técnica UCP.

O UCP apresenta algumas limitações que provocam dificuldades para quem deseja adotar a métrica. Estes problemas podem ter contribuído para que até o momento, ela não tenha alcançado a popularidade obtida pela FPA. A seguir estão listados alguns dos principais problemas avaliados.

5.1 Complexidade dos UCs

Na técnica de UCP, a complexidade do sistema é determinada em função da complexidade dos UCs e dos atores (vide Fórmula 4.13). Cada UC pode apenas ser classificado em simples, médio ou complexo, recebendo peso 1, 2 ou 3, respectivamente.

Sendo assim, um UC pode ser no máximo, 3 vezes maior que outro, independentemente do número de cenários, entidades, pré-condições apresentados por cada um deles. Eis um exemplo da dificuldade provocada, suponha a seguinte situação: Um UC com 3 entidades e um único cenário pode ser considerado apenas três vezes menor (peso 1) que outro com 30 entidades e 8 cenários, provocando uma falha na comparação de tamanho entre os dois.

5.2 Uma Única Medida

No caso da métrica UCP, só se obtém um valor para o sistema por completo, pois como os UCs só são avaliados segundo um peso, não é possível a obtenção de medidas para cada um, separadamente.

A obtenção do tamanho dos UCs isoladamente poderia permitir avaliações de prioridade e de viabilidade por parte da equipe e dos gerentes de projeto. Por exemplo, a partir do tamanho de alguns UCs poderia ser desejável estimar o tempo necessário para o desenvolvimento dos mesmos, com a estimativa em mãos, o gerente poderia decidir pela implementação dos UCs menores ou do conjunto daqueles que poderiam ser entregues em um determinado prazo imposto pela diretoria.

Não se tendo uma medida do tamanho de cada UC em separado, não é possível se obter as estimativas para cada UC, nem tampouco tomar-se decisões fundamentadas na métrica.

5.3 Pontos por Tamanho de Caso de Uso

A métrica proposta para reduzir as limitações citadas é denominada Pontos por Tamanho de Caso de Uso (USP - Use Case Size Points).

Como a métrica pode ser aplicada para cada UC separadamente, após obter-se o USP de um UC, é possível a obtenção de estimativas do tempo e custo para o desenvolvimento de um UC em particular e não apenas do sistema por completo.

O USP determina o tamanho da funcionalidade através da estrutura e das seções encontradas em um UC, contabilizando-se o número e peso dos cenários, pré e pós-condições, atores, etc.

O USP de um UC pode variar muito em relação ao USP de um outro, isso permite uma melhor avaliação do tamanho dos UCs, pois um UC pode ficar muito maior que outro dependendo da forma com que o sistema está particionado pelos UCs. Ao contrário da técnica UCP, na qual um UC poderia ser apenas 3 vezes maior que outro (Peso 3 x Peso 1).

A principal informação considerada pelo analista na classificação de uma seção do UC é o número de entidades encontradas naquela seção. Maiores informações sobre entidades podem ser encontradas na Seção 6.1.2.

Tabela 5.1: Complexidade dos Atores

Complexidade	Número de informações	Qtde. USP
Simples	<= 5	2
Média	6 a 10	4
Complexa	> 10	6

5.4 Calculando o USP

A seguir é apresentado um roteiro passo-a-passo de como calcular o USP de um sistema ou de apenas um UC.

5.4.1 Determine Quais UCs Serão Analisados

Primeiramente, é necessário determinar quais UCs deseja-se avaliar. Pode-se optar por medir o tamanho do sistema por completo (avaliando-se todos os UCs) ou de apenas um subgrupo.

5.4.2 Avaliando Cada UC

Para determinar o USP de cada UC, devem ser verificados o número e o tamanho de cada seção contida no mesmo. Sendo assim, será necessário avaliar cada seção isoladamente, para enfim chegar a um número final para o UC.

1. **Classificação dos atores:** Cada ator tem sua complexidade (CA) determinada segundo a quantidade de informações que transmite ou que recebe do UC, sendo classificado segundo a Tabela 5.1.

   O total de pontos da seção Atores, o TPA, é dado pela Fórmula 5.1.

$$TPA = \sum_{i=1}^{i=n} CA_i \tag{5.1}$$

   n - Nº. de Atores do Caso de Uso

2. **Classificação das Pré-Condições:** Cada Pré-Condição do UC deve ter sua complexidade (CPrC) classificada de acordo com o número de expressões lógicas testadas, conforme a Tabela 5.2. Pois a necessidade de se testar as pré-condições pode adicionar significativa complexidade ao UC.

Tabela 5.2: Complexidade das Pré-Condições

Complexidade	Expressões testadas	Qtde. USP
Simples	1	1
Média	2 ou 3	2
Complexa	> 3	3

Tabela 5.3: Complexidade dos Cenários

Complexidade	N°. Entidades + N°. Passos	Qtde. USP
Muito Simples	<= 5	4
Simples	6 a 10	6
Média	11 a 15	8
Complexa	16 a 20	12
Muito Complexa	> 20	16

Logo após, devem ser somadas as complexidades de todas as Pré-Condições do UC, formando o TPPrC (Total de Pontos das Pré-Condições), dado pela Fórmula 5.2.

$$TPPrC = \sum_{i=1}^{i=n} CPrC_i \tag{5.2}$$

n - N°. de Pré-Condições do UC

3. **Classificação do Cenário Principal:** A complexidade do Cenário Principal do UC deve ser classificada segundo a quantidade de entidades que o mesmo apresenta e o número de passos elementares necessários para a conclusão do cenário, as duas quantidades devem ser somadas (número de entidades + número de passos), o valor deve classificar o cenário segundo a Tabela 5.3. O número de pontos do cenário principal é referenciado pela sigla PCP.

4. **Classificação dos Cenários Alternativos:** De forma similar à classificação executada para o cenário principal, devem ser classificados cada um dos cenários alternativos (quanto à complexidade inerente àquele cenário) presentes no UC, cada cenário alternativo recebe um número de pontos (PCA).

   A complexidade dos cenários alternativos varia da mesma forma que a complexidade do cenário principal, conforme a Tabela 5.3. O total de pontos dos cenários alternativos é dado pela Fórmula 5.3.

Tabela 5.4: Complexidade das Exceções

Complexidade	Nº. Expressões Testadas	Qtde. USP
Simples	1	1
Média	2 ou 3	2
Complexa	> 3	3

Tabela 5.5: Complexidade das Pós-Condições

Complexidade	Nº. de Entidades	Qtde. USP
Simples	<= 3	1
Média	4 a 6	2
Complexa	> 6	3

$$TPCA = \sum_{i=1}^{i=n} PCA_i \tag{5.3}$$

n - Nº. de Cenários Alternativos do UC

5. **Classificação das Exceções**

   Cada exceção presente no UC também deve ser classificada segundo a sua complexidade (CE), determinada a partir do número de expressões lógicas testadas para detectar a ocorrência da exceção. O total de pontos adicionados pelas exceções é dado pela Fórmula 5.4.

   Esta classificação permitirá avaliar a complexidade de identificação (por parte do sistema) daquela exceção. A complexidade das exceções varia de acordo com a Tabela 5.4.

$$TPE = \sum_{i=1}^{i=n} CE_i \tag{5.4}$$

n - Nº. de Exceções do UC

6. **Classificação das Pós-Condições**

   A necessidade de se deixar o sistema no estado indicado pelas pós-condições pode adicionar complexidade ao UC. A complexidade das pós-condições varia de conforme a Tabela 5.5.

   A seguir, devem ser somadas as complexidades de todas as Pós-Condições do UC (CPoC), formando o TPPoC (Total de Pontos das Pós-Condições), dado pela Fórmula 5.5.

$$TPPoC = \sum_{i=1}^{i=n} CPoC_i \tag{5.5}$$

n - Nº. de Pós-Condições do Caso de Uso

7. **Determinando o valor não ajustado**

   Após determinar o número de pontos de cada seção, deve-se obter o número de pontos por tamanho não ajustado (UUSP-Unajusted Use-case Size Points).

   O total de pontos é obtido através do simples somatório do número de pontos encontrado em cada uma das seções avaliadas. Esse total é representado pela Fórmula 5.6.

$$UUSP = TPA + TPPrC + PCP + TPCA + TPE + TPPoc$$

8. **Aplicando-se os fatores de ajuste**

   Após obter-se o número de pontos não ajustado já se pode ter uma idéia do tamanho geral do sistema. O número de pontos não ajustado representa o tamanho da funcionalidade, desconsiderando as dificuldades e facilidades encontradas no desenvolvimento, que podem advir dos fatores técnicos e ambientais presentes no sofware e na organização.

   - **Fatores Técnicos de Ajuste** Os fatores técnicos representam a influência que determinadas características técnicas (presentes no software sendo desenvolvido e inerentes a todos os UCs do sistema), podem ter sobre o software.

     O valor do Fator Técnico de Ajuste é determinado pela soma da influência (I_i) de cada um dos fatores apresentados na Tabela 5.6. A influência dos fatores pode variar de 0 (sem importância) a 5 (item imprescindível).

     A seguir, a fórmula de cálculo do FTA, Fórmula 5.6.

$$FTA = 0.65 + (0.01 * \sum_{i=1}^{i=14} I_i) \tag{5.6}$$

Tabela 5.6: Peso de Fatores Técnicos, adaptada da Tabela 4.2

Fator	Requisito	Influência
F1	Comunicação de dados	I1
F2	Processamento Distribuído	I2
F3	Desempenho	I3
F4	Utilização do equipamento	I4
F5	Volume de transações	I5
F6	Entrada de dados on-line	I6
F7	Eficiência do usuário final	I7
F8	Atualização on-line	I8
F9	Reutilização de código-fonte	I9
F10	Processamento complexo	I10
F11	Facilidade de implantação	I11
F12	Facilidade Operacional	I12
F13	Multiplicidade de locais	I13
F14	Facilidade de mudanças	I14

- **Fatores Ambientais de Ajuste**

  Os fatores ambientais representam de que forma podem influenciar no custo do software, determinadas características presentes no ambiente de desenvolvimento.

  O Fator Ambiental de Ajuste é determinado pela Fórmula 5.7, que considera as características enumeradas pela Tabela 5.7. A influência dos fatores ambientais também pode variar de 0 (fator ausente ou sem influência) a 5 (fator presente em todas as etapas do projeto).

Tabela 5.7: Pesos de Fatores Ambientais, adaptada de [16]

Fator	Descrição	Influência
E1	Existência de processo formal de desenvolvimento	I1
E2	Experiência com a aplicação em desenvolvimento	I2
E3	Experiência da equipe com as tecnologias utilizadas	I3
E4	Presença de analista experiente	I4
E5	Requisitos estáveis	I5

$$FAA = (0.01 * \sum_{i=1}^{i=5} I_i) \tag{5.7}$$

9. **Finalizando o cálculo**

   Finalmente o valor final de USP de um caso de uso é determinado pela Fórmula 5.8.

$$USP = UUSP * (FTA - FAA) \tag{5.8}$$

No caso de um conjunto de UCs, o cálculo dos fatores de ajuste deve ser realizado uma única vez, utilizando-se os valores obtidos para ajustar cada UC. Para obter-se o USP de sistema ou módulo completo, basta realizar uma soma simples do USP de cada UC pertencente ao mesmo.

O valor não ajustado é importante para, entre outras coisas, se comparar projetos de organizações diferentes, pois como os fatores de ajuste podem ser particulares a uma determinada organização, o valor final pode estar bastante influenciado por um fator particular a uma das organizações ou pela subjetividade empregada na análise de influência desses fatores.

Logo abaixo, um exemplo de como o valor não ajustado pode ser importante:

Ex 1: O UUSP de dois projetos é de 1000 pontos.

Uma organização aplica determinado fator de ajuste ambiental presente em seu processo de desenvolvimento, reduzindo o valor final para 800 USP.

A outra organização também aplica um fator de ajuste presente em seu processo que reduz o valor final para 950 USP.

A pergunta é: Qual projeto é maior, o de 800 USP ou ou de 950 USP?

Na verdade eles são projetos de mesmo tamanho (1000 pontos não ajustados), o que faz com que os valores finais sejam diferentes é a diferença entre os processos de desenvolvimento das duas organizações, pois o fator que influencia a primeira organização é mais impactante que o fator da 2a. organização.

Para efetuar esse tipo de comparações, o valor não ajustado (UUSP) se mostra mais adequado.

5.4.3 Exemplo de Aplicação do USP

Para demonstrar a aplicação da técnica, a seguir será efetuado o cálculo de USP para o caso de uso "Efetuando Saque de Dinheiro", que foi introduzido no Capítulo 2 e que está novamente apresentado na Figura 5.1.

1. **Classificação dos atores**

   - Cliente do Banco - Simples (3 informações) - 2 UUSP

Caso de Uso: Efetuando saque de dinheiro

Atores: Cliente do banco

Pré-Condições: 1. O cliente deve estar cadastrado no banco; 2. O cliente deve ter saldo disponível; 3. O caixa automático deve estar ligado e aguardando operação.

Cenário Principal:
1. O cliente chega até o caixa automático desejando sacar uma quantia em dinheiro.
2. O sistema do caixa automático está aguardando operação, emitindo uma mensagem que solicita ao cliente que insira seu cartão do banco.
3. O cliente insere o cartão do banco, o sistema valida o cartão e solicita a senha.
4. O cliente informa a sua senha.
5. O sistema valida a senha e fornece um menu com as seguintes opções: Consultar saldo, Efetuar saque, Sair.
6. O cliente seleciona Efetuar saque.
7. O sistema permite ao usuário a escolha do valor a sacar.
8. O cliente informa o valor a sacar e confirma a operação.
9. O sistema verifica o saldo, efetua o saque, entrega o dinheiro e diminui o saldo do cliente conforme o valor sacado.

Cursos Alternativos:
6A. O cliente escolhe Consultar saldo
6A1. O sistema emite um comprovante informando o saldo do cliente
6B. O cliente escolhe Sair
6B1. O sistema encerra a operação e devolve o cartão ao cliente.

Exceções
3A. O cartão inserido não pertence ao banco ou está com problemas.
3A1. O sistema emite mensagem informando o cliente sobre o problema ocorrido e volta a aguardar operação.
5A. A senha informada é inválida
5A1. O sistema emite mensagem informando o cliente sobre o problema ocorrido, solicita a senha novamente,até um máximo de três vezes.
9A1. O saldo é insuficiente para permitir o saque solicitado pelo usuário.
9A2. O sistema informa ao cliente que o saldo (valor do saldo) é insuficiente para completar o saque e volta a solicitar a operação desejada, passo 5.

Pós Condições: 1. O sistema retornou ao estado aguardando operação.

Figura 5.1: Um Caso de Uso Completo

$$TPA = 2 \tag{5.9}$$

2. **Classificação das Pré-Condições**

   - Cadastro do Cliente - Simples (1 expressão lógica) - 1 UUSP
   - Disponibilidade de Saldo - Simples (1 expressão lógica)- 1 UUSP
   - Estado do caixa automático - Simples (1 expressão lógica)- 1 UUSP

$$TPPrC = 1 + 1 + 1 = 3 \tag{5.10}$$

3. **Classificação do Cenário Principal**

   O cenário principal é composto de dez entidades: cliente, caixa automático, saque, dinheiro, quantia, cartão, banco, senha, mensagem, saldo. O cenário também está dividido em 9 passos. Somando-se obtém-se o valor 19.

   Segundo a Tabela 5.3, o cenário principal é classificado como Complexo, correspondendo a 12 UUSP.

$$PCP = 12 \tag{5.11}$$

4. **Classificação dos Cenários Alternativos**

   - Consultar Saldo - (4 ent. + 2 passos) - Simples - 6 UUSP
   - Saída do Sistema - (3 ent. + 2 passos) - Muito Simples - 4 UUSP

$$TPCA = 6 + 4 = 10 \tag{5.12}$$

5. **Classificação das Exceções**

   - Falha no cartão - Média (2 expressões lógicas) - 2 UUSP
   - Senha inválida - Simples (1 expressão lógica) - 1 UUSP
   - Saldo insuficiente - Simples (1 expressão lógica) - 1 UUSP

$$TPE = 2 + 1 + 1 = 4 \tag{5.13}$$

6. **Classificação das Pós-Condições**

   - Estado final do sistema - Simples (3 entidades) - 1 UUSP

$$TPPoC = 1 \tag{5.14}$$

7. **Determinando o valor não ajustado**

$$UUSP = TPA + TPPrC + PCP + TPCA + TPE + TPPoc \tag{5.15}$$

$$UUSP = 2 + 3 + 12 + 10 + 4 + 1 = 32 \tag{5.16}$$

8. **Aplicando-se os fatores de ajuste**

   - **Fator Técnico de Ajuste** Os fatores técnicos que possuem influência sobre o sistema são:
     - F9 - Código reusável - I9 = 1
     - F10 - Processamento Complexo - I10 = 1
     - F11 - Facilidade de Implantação - I11 = 1

$$FTA = 0.65 + (0.01 * (I9 + I10 + I11)) \tag{5.17}$$

$$FTA = 0.65 + (0.01 * (1 + 1 + 1)) = 0.68 \tag{5.18}$$

   - **Fatores Ambientais de Ajuste**
     Os fatores ambientais presentes no desenvolvimento do sofware são:
     - E4 - Presença de analista experiente - I4 = 1
     - E5 - Requisitos estáveis - I5 = 2

$$FAA = 0.01 * (I4 + I5)) \tag{5.19}$$

$$FAA = 0.01 * (1 + 2) = 0.03 \tag{5.20}$$

9. **Finalizando o cálculo**

$$USP = UUSP * (FTA - FAA) = 32 * (0.68 - 0.03) = 20.8 \quad (5.21)$$

O número final de USP para um sistema hipotético que continha apenas o caso de uso "Efetuando Saque de Dinheiro" é aproximadamente 20.8.

5.5 Pontos por Tamanho de Caso de Uso *Fuzzy*

Algumas características e limitações da técnica de FPA motivaram Lima Júnior [21] a estender a técnica de Análise de Pontos por Função (Seção 4.5.3 do Capítulo 4) utilizando-se da teoria dos Conjuntos *Fuzzy*, criando assim a Análise de Pontos por Função *Fuzzy* (FFPA-Fuzzy Function Point Analysis).

Algumas das limitações observadas devem-se à existência de tabelas de classificação de funcionalidades, que fazem a correspondência entre variáveis lingüísticas de complexidade e seus respectivos valores, os quais definem o tamanho de tais funcionalidades. A utilização de tais tabelas não permite uma mudança gradual de uma faixa de complexidade para outra.

Embora o USP seja proveniente da técnica de UCP e apresente algumas inovações, ele da mesma maneira que seu antecessor e que a FPA também utiliza uma classificação discreta da complexidade das funcionalidades, apresentando os mesmos problemas e limitações.

Sendo assim, as vantagens decorrentes da aplicação da teoria *Fuzzy* sobre a técnica FPA também podem ser obtidas com a extensão do modelo original do USP, aqui apresentado.

O modelo estendido do USP será denominado Pontos por Tamanho de Caso de Uso *Fuzzy* (*Fuzzy* Use case Size Points-USPF), e sua construção será detalhada a seguir.

5.5.1 Estendendo o USP em USPF

Conforme apresentado no Capítulo 4, a extensão da FPA em FFPA [21] foi realizada pelo seguinte conjunto de passos:

1. Fuzificação dos termos lingüísticos das matrizes de complexidade da FPA, através da geração de números *Fuzzy* (trapezoidais);

2. Extensão das matrizes de complexidade da FPA, gerando-se novos termos lingüísticos;

3. Determinação do valor em pontos por função dos novos termos lingüísticos, gerados na etapa anterior; e
4. Defuzificação dos valores dos termos lingüísticos da FFPA em pontos por função.

Como os passos intermediários (2 e 3) não são considerados essenciais para a extensão do USP em USPF (Seção 4.5.3), ela será realizada através de apenas 2 passos, são eles:

1. Fuzificação dos termos linguísticos
2. Defuzificação dos termos linguísticos

5.5.2 Fuzificação dos Termos Lingüísticos

A etapa de fuzificação transformará as tabelas de classificação de complexidade em uma classificação contínua. Isso será realizado através da geração de um número *Fuzzy* trapezoidal para cada faixa de complexidade encontrada nas diversas tabelas de classificação de complexidade. Assim cada tipo de seção de caso de uso (atores, pós-condições, etc) será classificado através de um gráfico contendo um número *Fuzzy* trapezoidal para cada faixa de complexidade pré-existente.

As seções seguintes descrevem com mais detalhes a criação dos números trapezoidais a partir das variáveis lingüísticas, que foram gerados da seguinte maneira [21]:

"O valor de m_i assume o limite inferior do termo lingüístico i da matriz de complexidade considerada. O valor de n_i é calculado a partir da média aritmética entre os valores de m_i e m_{i+1}, sendo que este resultado deve ser inteiro e arredondado. O valor de n_{i+1} e o valor de m_{i+1} foram atribuídos a a_i e b_i, respectivamente. Alguns ajustes são realizados quando se tratam do primeiro ou último termo lingüístico". Resumidamente:

- m_i = limite inferior do termo lingüístico T_i na tabela de classificação
- $n_i = \frac{(m_i + m_{i+1})}{2}$
- $a_i = n_{i-1}$
- $b_i = m_{i+1}$

No caso do USPF, o arredondamento não foi realizado visando preservar integralmente a característica de continuidade dos intervalos gerados.

Tabela de Classificação dos Atores

A forma original de classificação de atores está apresentada na Tabela 5.1. Os números *Fuzzy* gerados para os termos Simples, Média e Complexa que estão presentes na tabela foram obtidos da seguinte maneira:

- m1 = 1
- n1 = (1+6)/2 = 3.5
- $a1 = \not\exists$
- b1 = 6
- m2 = 6
- n2 = (6+11)/2 = 8.5
- a2 = 3.5
- b2 = 11
- m3 = 11
- $n3 = \not\exists$
- a3 = 8.5
- $b3 = \not\exists$

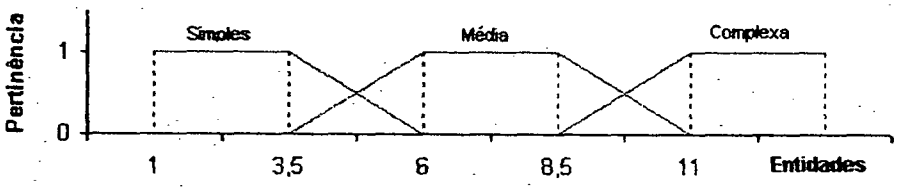

Figura 5.2: Números *Fuzzy* Correspondentes à Tabela de Classificação de Atores

Tabela de Classificação das Pré-Condições

A forma original de classificação de pré-condições está apresentada na Tabela 5.2. Os números *Fuzzy* gerados para os termos Simples, Média e Complexa que estão presentes na tabela foram obtidos da seguinte maneira:

- m1 = 1
- n1 = (1+2)/2 = 1.5
- $a1 = \not\exists$
- b1 = 2
- m2 = 2
- n2 = (2+4)/2 = 3
- a2 = 1.5
- b2 = 4
- m3 = 4
- $n3 = \not\exists$
- a3 = 3
- $b3 = \not\exists$

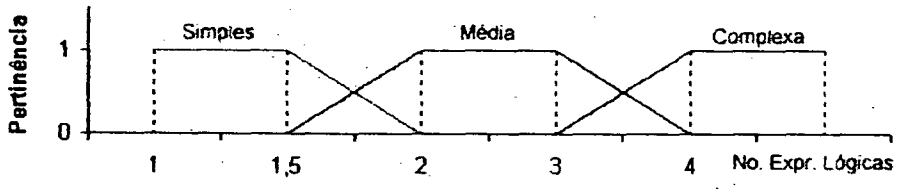

Figura 5.3: Números *Fuzzy* Correspondentes à Tabela de Classificação de Pré-Condições

Tabela de Classificação dos Cenários

A forma original de classificação de cenários está apresentada na Tabela 5.3. Os números *Fuzzy* gerados para os termos Muito Simples, Simples, Média, Complexa e Muito Complexa que estão presentes na tabela foram obtidos da seguinte maneira:

- m1 = 1
- n1 = (1+6)/2 = 3.5
- a1 = $\nexists$
- b1 = 6
- m2 = 6
- n2 = (6+11)/2 = 8.5
- a2 = 3.5
- b2 = 11
- m3 = 11
- n3 = (11+16)/2 = 13.5
- a3 = 8.5
- b3 = 16
- m4 = 16
- n4 = (16+21)/2 = 18.5
- a4 = 13.5
- b4 = 21
- m5 = 21
- n5 = $\nexists$
- a5 = 18.5
- b5 = $\nexists$

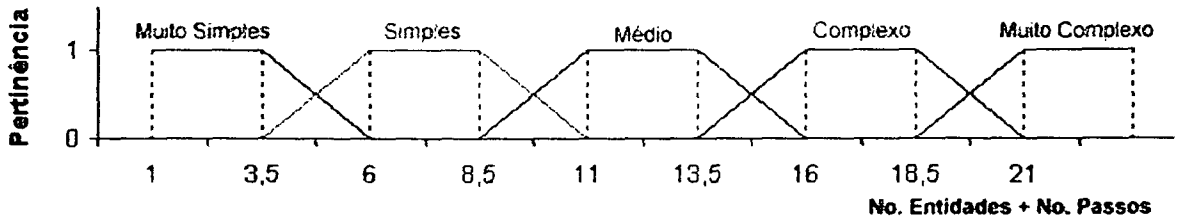

Figura 5.4: Números *Fuzzy* Correspondentes à Tabela de Classificação de Cenários

Tabela de Classificação das Exceções

A forma original de classificação de exceções está apresentada na Tabela 5.4. Os números *Fuzzy* gerados para os termos Simples, Média e Complexa que estão presentes na tabela foram obtidos da seguinte maneira:

- m1 = 1
- n1 = (1+2)/2 = 1.5
- $a1 = \not\exists$
- b1 = 2
- m2 = 2
- n2 = (2+4)/2 = 3
- a2 = 1.5
- b2 = 4
- m3 = 4
- $n3 = \not\exists$
- a3 = 3
- $b3 = \not\exists$

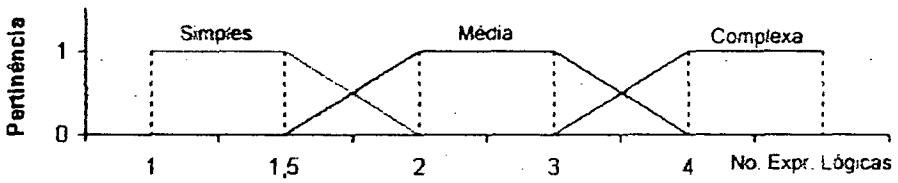

Figura 5.5: Números *Fuzzy* Correspondentes à Tabela de Classificação de Exceções

Tabela de Classificação das Pós-Condições

A forma original de classificação de pós-condições está apresentada na Tabela 5.5. Os números *Fuzzy* gerados para os termos Simples, Média e Complexa que estão presentes na tabela foram obtidos da seguinte maneira:

- m1 = 1
- n1 = (1+4)/2 = 2.5
- a1 = $\not\exists$
- b1 = 4
- m2 = 4
- n2 = (4+7)/2 = 5.5
- a2 = 2.5
- b2 = 7
- m3 = 7
- n3 = $\not\exists$
- a3 = 5.5
- b3 = $\not\exists$

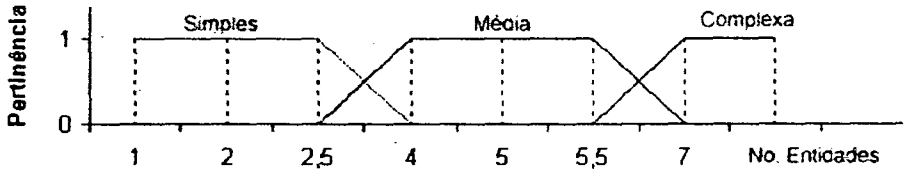

Figura 5.6: Números *Fuzzy* Correspondentes à Tabela de Classificação de Pós-Condições

5.5.3 Defuzificação dos Termos Lingüísticos

Assim como a estratégia de fuzificação, a defuzificação também foi a mesma que a adotada por [21]. O processo de defuzificação visa transformar os termos linguísticos da USPF em um valor em UUSPF, não sujeito às mudanças abruptas de classe existentes no USP simples.

A defuzificação foi realizada através da aplicação de duas regras simples. Depois disto, o UUSPF (USPF não ajustado) pode ser calculado. O processo envolve o cálculo da função de pertinência $\mu(x)$, que representa o quanto o elemento "x" (um número real) pertence ao conjunto em questão (neste caso, o grau de pertinência do número a uma faixa de complexidade).

Cada uma das regras é aplicada a um tipo de situação, a primeira é quando o número obtido pertence a um único número *Fuzzy* e a última é quando o valor está entre dois números *Fuzzy* (em uma região de transição).

1. Se o número a ser classificado (expressões lógicas, entidades, entidades + passos, etc) estiver entre os valores m_i e n_i do número *Fuzzy* correspondente, nesse caso o valor em pontos será o valor em pontos da faixa a que pertence este número. Isto ocorre porque ele pertence à base superior do número trapezoidal. Neste caso, o valor da função de pertinência $\mu(x)$ é igual a um (1 ou 100% de pertinência), ou seja, o mesmo valor que o USP convencional forneceria.

2. Se o número x a ser classificado (expressões lógicas, entidades, entidades + passos, etc) estiver entre os valores de n_i e b_i do número *Fuzzy* correspondente, ou seja, está entre a faixa de valores comum a dois números *Fuzzy* (pois também está entre a_{i+1} e m_{i+1}), será necessário calcular o grau de pertinência do número para cada um dos números *Fuzzy* correspondentes (ou o grau de pertinência à faixa de complexidade que cada um representa), realizado através das Fórmulas 5.22 e 5.23. Após isso, calcula-se a defuzificação, Fórmula 5.24.

$$\mu(x) = (b_i - x)/(b_i - n_i) \quad (5.22) \qquad \overline{\mu}(x) = 1 - \mu(x) \quad (5.23)$$

$$dUUSPF(x) = \overline{\mu}(x).UUSP_i + \overline{\mu}(x).UUSP_{i+1} \quad (5.24)$$

Onde dUUSPF(x) é o valor obtido após a defuzificação do número "x", $UUSP_i$ é o valor de UUSP para a faixa "i", $UUSP_{(i+1)}$ é o valor de UUSP para a faixa subseqüente, e $\mu(x)$ e $\overline{\mu}(x)$ são os graus de pertinência à cada uma das faixas, respectivamente.

5.5.4 Exemplo de Aplicação do USPF

Para demonstrar a aplicação da técnica, a seguir será efetuado o cálculo de USP para o caso de uso "Efetuando Saque de Dinheiro", que foi introduzido no Capítulo 2 e que está novamente apresentado na Figura 5.1.

1. **Classificação dos atores**

   - Cliente do Banco - 3 informações

   Segundo o gráfico da Figura 5.2, um ator com até 3 informações corresponde a 2 UUSPF (Ator Simples), pois o valor 3 (número de informações) encontra-se na base superior do número fuzzy trapezoidal.

$$TPA = 2 \quad (5.25)$$

2. **Classificação das Pré-Condições**

   - Cadastro do Cliente - 1 expressão lógica
   - Disponibilidade de Saldo - 1 expressão lógica
   - Estado do caixa automático - 1 expressão lógica

     As 3 Pré-Condições listadas acima correspondem a 1 UUSPF cada, pois o valor 1 (expressão lógica) encontra-se na base superior do número fuzzy trapezoidal de Pré-Condições.

$$TPPrC = 1 + 1 + 1 = 3 \quad (5.26)$$

3. **Classificação do Cenário Principal**

   O cenário principal é composto de dez entidades: cliente, caixa automático, saque, dinheiro, quantia, cartão, banco, senha, mensagem, saldo.

   O cenário também é dividido em 9 passos, logo o valor a ser considerado será 19 (10 entidades + 9 passos).

   Conforme verifica-se pela Figura 5.4 que apresenta o número fuzzy correspondente à classificação de cenários, o valor 19 encontra-se na intersecção entre as faixas de complexidade Alta e Muito Alta. Sendo assim, será necessário aplicar a Fórmula de defuzificação, conforme definida em 5.24.

   Cenário Principal - 19 (10 entidades + 9 passos)

$$\mu(x) = (21 - 19)/(21 - 18.5) = 0.8 \tag{5.27}$$

$$\overline{\mu}(x) = 1 - 0.8 = 0.2 \tag{5.28}$$

$$p_d = 0.8 * 12 + 0.2 * 16 = 12.8 \tag{5.29}$$

$$PCP = 12.8 \tag{5.30}$$

4. **Classificação dos Cenários Alternativos**

   - Consultar Saldo - 6 (4 entidades + 2 passos)
     O valor 6 pertence a base superior da faixa de complexidade Simples, recebendo portanto 6 UUSPF.
   - Saída do Sistema - 5 (3 entidades + 2 passos)
     Novamente se faz necessário o processo de defuzificação, conforme a Fórmula 5.24.

$$\mu(x) = (6 - 5)/(6 - 3.5) = 0.4 \tag{5.31}$$

$$\overline{\mu}(x) = 1 - 0.4 = 0.6 \tag{5.32}$$

$$p_d = 0.4 * 4 + 0.6 * 6 = 5.2 \tag{5.33}$$

$$TPCA = 6 + 5.2 = 11.2 \tag{5.34}$$

5. **Classificação das Exceções**

- Falha no cartão - 2 expressões lógicas Conforme a Figura 5.5, corresponde a 2 UUSPF.
- Senha inválida - 1 expressão lógica Conforme a Figura 5.5, corresponde a 1 UUSPF.
- Saldo insuficiente - 1 expressão lógica Conforme a Figura 5.5, corresponde a 1 UUSPF.

$$TPE = 2 + 1 + 1 = 4 \tag{5.35}$$

6. **Classificação das Pós-Condições**

   - Estado final do sistema - 3 entidades

$$\mu(x) = (4 - 3)/(4 - 2.5) = 0.66 \tag{5.36}$$
$$\overline{\mu}(x) = 1 - 0.66 = 0.34 \tag{5.37}$$
$$p_d = 0.66 * 1 + 0.34 * 2 = 1.34 \tag{5.38}$$

$$TPPoC = 1.34 \tag{5.39}$$

7. **Determinando o valor não ajustado**

$$UUSPF = TPA + TPPrC + PCP + TPCA + TPE + TPPoC \tag{5.40}$$

$$UUSPF = 2 + 3 + 12.8 + 11.2 + 4 + 1.34 = 34.34 \tag{5.41}$$

8. **Finalizando o cálculo**

   Como para ambas as técnicas (USP e USPF) o processo de ajuste de valor ocorre da mesma forma, serão reutilizados os valores do FTA e FAA calculados no exemplo de utilização do USP, 0.6325 e 1.325 respectivamente.

$$USPF = UUSPF * (FTA - FAA) = 33.34 * (0.68 - 0.03) = 21.67 \tag{5.42}$$

O número final de USPF para um sistema hipotético que continha apenas o caso de uso "Efetuando Saque de Dinheiro" é aproximadamente 21.67.

5.6 Considerações Finais sobre o USP e o USPF

A métrica proposta, o USP, busca maior coerência visando determinar o tamanho do caso de uso através de uma análise de suas seções internas. Ao efetuar este tipo de análise, a técnica dá ao analista responsável maiores chances de sucesso na tarefa de avaliar o UC, pois o mesmo não precisa mais determinar sua complexidade em um único passo, mas sim através do particionamento dessa tarefa em pequenas avaliações, seção por seção. Falando de software, particionar é sempre a chave para reduzir a complexidade, qualquer que seja a tarefa a ser desempenhada.

O USPF traz os conceitos e teorias largamente utilizados dos conjuntos Fuzzy para o contexto das métricas de software, ao empregar números trapezoidais para modelar as faixas de complexidade das tabelas de classificação da métrica, a variante Fuzzy permite uma mensuração mais adequada dos casos de uso. No entanto, esta variação agrega complexidade ao USP, que prima muito pela sua simplicidade. Neste caso, ferramentas de automatização do processo de medição desempenhariam um papel fundamental na adoção da métrica. A coleta automática do USP e USPF é discutida no próximo capítulo.

Ambas as métricas trazem possibilidades de uma mensuração direta de casos de uso, limitando a necessidade de transposição de paradigmas durante o processo de medição e assim ampliando as chances de sucesso na execução de tal atividade.

Capítulo 6

Automatizando a coleta do USP e USPF

Geralmente, a aplicação do UCP é realizada de forma manual. Na maioria das vezes, o analista deve analisar UCs desenvolvidos por outras pessoas e, classificar cada UC e ator do sistema segundo um peso, para efetuar essa atribuição o analista precisa contar manualmente o número de transações ou entidades. Após a determinação do peso dos UCs, estes devem ser somados para a obtenção do valor não ajustado. A aplicação dos fatores de ajuste também normalmente é realizada de forma manual.

De forma similar ao exposto no parágrafo anterior, a aplicação da primeira métrica proposta (USP) também é manual. Ela requer a avaliação e contagem manual dos pesos de diversas seções de um UC. A aplicação dos fatores de ajuste é, igualmente, manual. Na técnica USPF, a aplicação dos conceitos *Fuzzy* nas faixas de complexidade, adiciona complexidade ao processo de medição.

Dessa forma, o objetivo desse capítulo é apresentar uma notação estruturada e um protótipo, ambos desenvolvidos visando à automatização do processo de medição do USP e do USPF.

6.1 Notação em XML

Conforme descrito nos capítulos anteriores, a inexistência de uma notação padrão para a escrita de casos de uso provoca uma certa confusão para quem deseja iniciar a utilização da técnica, principalmente para a utilização de métricas relacionadas a casos de uso.

A linguagem de marcação extensível (eXtended Markup Language-XML) desenvolvida pelo consórcio W3C, está sendo adotada como o padrão para

troca de documentos por quase todas as grandes empresas de tecnologia da informação. A adoção em larga escala deve-se ao fato do XML ser um modelo simples e textual, bastante similar à linguagem de construção de páginas Web, o HTML [6].

Pelos motivos mencionados acima, o XML é de fácil entendimento, sendo possível a um usuário abrir o arquivo e entender o que está sendo transmitido, essa facilidade não é encontrada em modelos convencionais de intercâmbio de informações de software [6].

Esse capítulo propõe uma notação estruturada para a escrita e principalmente para o armazenamento de casos de uso em formato XML, o que permitiria já a partir do término da especificação, a obtenção automática de medidas do tamanho do UC a ser desenvolvido.

A construção de um caso de uso diretamente em formato XML pode não ser tão simples quanto redigir um texto comum, por isso, se faz necessário o uso de ferramentas adequadas que transformem o texto de um caso de uso em formato XML.

O armazenamento em formato XML pressupõe a utilização de casos de uso estruturados, ou seja, aqueles que são escritos utilizando-se das seções e terminologias que a UML prevê.

A proposta combina em um único documento, a especificação estruturada dos requisitos e uma base de informações para a extração de métricas. A aplicação da notação também possibilita a padronização e uniformização da documentação gerada, pois os UCs são descritos de maneira uniforme, promovendo assim, todos aqueles benefícios decorrentes de uma padronização: aumento de produtividade, documentação similar, intercâmbio entre ferramentas, etc.

Também se deseja permitir a identificação de elementos necessários à classificação da complexidade das seções do UC, elementos que não estão presentes na notação tradicional de UCs.

As próximas seções apresentam os principais elementos do formato proposto.

6.1.1 XML

O XML é um formato de texto simples, muito flexível, derivado do SGML. Originalmente projetado para atender aos desafios da publicação eletrônica em larga escala, XML também está desempenhando um papel importante e crescente na troca de informações e dados dos mais variados tipos, na Web e em outros diversos lugares [14].

O consórcio responsável pela manutenção e ampliação do XML, é o W3C

(World Wide Web Consortium), o mesmo é formado pelas principais empresas e organizações ligadas a software.

A composição de um documento XML é determinada pelas suas "tags", que são na verdade instruções de como o documento deve ser interpretado, elas delimitam e organizam o conteúdo de dados que estão sendo representados pelo XML.

Veja um exemplo simples de XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENTO>
    <TEXTO>
        Esse é um documento XML.
    </TEXTO>
    <UNIVERSIDADE>
        Universidade Federal do Paraná - UFPR
    </UNIVERSIDADE>
</DOCUMENTO>
```

A 1a. linha do documento anterior apresenta informações sobre a versão de XML que está sendo utilizada e o tipo de codificação de caracteres utilizado, informações que podem ser importantes para o processamento e manipulação do documento. Além disso, o documento é composto de 3 tags (DOCUMENTO, TEXTO, UNIVERSIDADE).

A tag DOCUMENTO apenas delimita as outras duas seções, TEXTO e UNIVERSIDADE, já estas outras duas trazem uma informação cada. É notória a simplicidade do documento e a liberdade com que se pode estruturar a informação.

"O XML tornou-se uma sintaxe universal para descrever e estruturar dados independentemente da lógica da aplicação. O XML pode ser usado para definir um número ilimitado de linguagens específicas para indústrias e aplicações. O XML promete simplificar e reduzir o custo de intercâmbio de dados e publicação no ambiente Web."... "O XML oferece portabilidade e reutilização sobre diferentes plataformas e dispositivos. É também flexível e extensível, permitindo que novas tags sejam adicionadas sem quebrar a estrutura pré-existente de um documento. Baseado no Unicode, o XML provê suporte global a linguagens."[8, 41].

Conforme pode-se observar, o XML está disseminado pelas organizações no mundo todo, o formato é utilizado tanto no ambiente acadêmico quanto no empresarial. O XML apresenta-se adequado para efetuar o armazenamento de qualquer tipo de informação, incluindo aí os UCs.

As próximas seções descreverão uma proposta de notação padrão para UCs e seu respectivo armazenamento em linguagem XML.

6.1.2 Entidades

Nesta nova notação, as entidades desempenham um papel especial, pois o número de entidades existentes em algumas seções dos UCs fornece subsídio importante na determinação da complexidade de cada uma das seções.

Como apresentado anteriormente, Symon [42] define uma entidade assim: "Qualquer coisa (objeto, real ou abstrato) do mundo real sobre o qual o sistema provê informação".

As entidades reveladas em um UC tendem a ter efeitos sobre o projeto e sobre a implementação da funcionalidade descrita pelo UC. Uma entidade pode vir a demandar uma estrutura de dados, armazenamento de informação, enfim, as entidades potencialmente aumentam a complexidade da funcionalidade na qual elas estão inseridas.

Para que as entidades possam ser identificadas, está prevista uma notação para que o escritor do UC possa destacá-las dentro do texto. O autor do UC deverá colocar as entidades entre colchetes, conforme o exemplo que será apresentado a seguir, onde foram identificadas 5 entidades: "itens", "compra", "caixa", "supermercado" e "item".

```
<CENARIO_PRINCIPAL>
    <DESCRICAO>
       #1. O [cliente] entrega os [itens] da [compra] ao
       [caixa] do [supermercado].
       #2. O caixa passa [item] por item da compra.
    </DESCRICAO>
</CENARIO_PRINCIPAL>
```

O autor do UC deve utilizar de bom senso ao denotar um vocábulo como entidade, pois eventualmente, podem existir entidades que não precisam ser denotadas, devido a não adicionarem nenhuma espécie de complexidade ou restrição ao UC.

Abaixo, algumas práticas recomendadas para identificar entidades:

- Atores são considerados entidades.
- O sistema sendo modelado não deve ser considerado uma entidade.
- Características (atributos) das entidades não devem ser consideradas como entidades, embora sejam substantivos. Ex: nome, descrição, idade, cor, valor, etc.
- Termos comumente utilizados em casos de uso e em descrições de requisitos, mas que não pertencem ao domínio do problema, não devem

ser considerados como entidades. Ex: Operação, funcionalidade, informações, detalhe, etc.

6.1.3 Estruturação do UC

A notação propõe uma forma de estruturar os UCs descritos com a mesma, dessa forma, objetiva minimizar as diferenças existentes entre os UCs, facilitando a tarefa de medí-los.

Os próximos sub-tópicos apresentam as seções que farão parte de UC, organizando-o e particionando-o.

Nome

Um nome significativo para o UC deve ser criado nesta seção, um texto simples que sintetize seu conteúdo, o nome do caso de uso é delimitado pela tag <CASO_DE_USO>.

Resumo

Esta seção visa fornecer ao leitor do caso de uso uma visão geral e sucinta dos propósitos da funcionalidade representada pelo documento. O resumo facilita a leitura e o entendimento da documentação. O resumo deve ter informação suficiente para permitir ao leitor decidir se é necessário ler o caso de uso por completo ou não. O resumo será delimitado pela tag <RESUMO>.

Título das Seções

É conveniente nomear e numerar os títulos das seções com um código para permitir a sua referenciação dentro do texto, isso facilita a organização e leitura do UC.

Atores

Na seção de Atores é apresentada uma listagem dos atores do caso de uso, a listagem está delimitada pela tag <ATORES>, e cada ator definido no UC será delimitado por uma tag <ATOR> e definido através do seguinte conjunto de informações:

- Nome do ator (<NOME>): Nome que melhor descreva o papel desempenhado pelo mesmo.

- Descritivo do ator (<DESCRICAO>): Texto que explique o papel do ator para com o sistema, suas principais características e peculiaridades, além de outras informações que sejam necessárias para a compreensão do mesmo.

- Número de informações (<NUM_INF>): Corresponde ao número de informações passadas pelo ator para o UC mais o número de informações recebidas pelo ator.

Veja um exemplo de definição de um ator:

```
<ATORES>
    <ATOR>
        <NOME> AT-01 Cliente externo </NOME>
        <DESCRICAO>
        O cliente externo utiliza o sistema para informar
        os dados de seu cartão de crédito, com isso conseguirá
        efetuar as suas compras. Normalmente é um usuário
        desprovido de qualquer conhecimento técnico.
        </DESCRICAO>
        <NUM_INF> 16 </NUM_INF>
    </ATOR>
</ATORES>
```

Pré-Condições

Nesta seção é apresentada uma listagem das pré-condições para que o caso de uso seja iniciado. A listagem deverá estar delimitada pela tag <PRE_CONDICOES>.

Cada pré-condição deve ser delimitada por uma tag <PRE_CONDICAO> e ser descrita da forma:

- Título (<TITULO>): Texto com poucas palavras (no máximo 10) que sintetize a descrição da pré-condição.

- Descritivo da pré-condição (<DESCRICAO>): Texto que explique a pré-condição e sua necessidade. Cada expressão lógica existente dentro do texto da pré-condição deve estar entre colchetes ([Expressão]).

Um exemplo de pré-condição pode ser observado a seguir:

```
<PRE_CONDICOES>
    <PRE_CONDICAO>
        <TITULO>
            PREC-01 - Validação de usuário/senha
        </TITULO>
        <DESCRICAO>
            [O usuário deverá ter passado pelo processo de
            login], que consiste em informar um nome de
            usuário e senha válidos no sistema.
        </DESCRICAO>
    </PRE_CONDICAO>
</PRE_CONDICOES>
```

Cenário Principal

O cenário principal deverá conter um texto que descreva, em uma seqüência de passos, a principal tarefa realizada pelo ator, ou seja, a principal operação do caso de uso. A tag que delimita todas as informações referentes ao cenário principal é <CENARIO_PRINCIPAL>.

O cenário principal também terá um indicador de complexidade que permitirá às métricas, classificar a dificuldade de implementação deste cenário. O cenário principal deve ser descrito da seguinte forma:

- Descritivo do cenário (<DESCRICAO>): Texto que descreva o cenário principal. Essa descrição deve estar organizada em pequenos passos dispostos em seqüência. A descrição de cada passo não deve superar mais que uma ou duas linhas. Os passos devem estar numerados e precedidos do caracter "#". As entidades presentes no texto também devem estar denotadas entre colchetes ([Entidade]).

Um exemplo de cenário principal pode ser observado a seguir:

```
<CENARIO_PRINCIPAL>
    <TITULO> Compra de itens à vista </TITULO>
    <DESCRICAO>
        #1. O [cliente] entrega os [itens] da [compra] ao
        [caixa do supermercado].
        #2. O caixa passa [item] por item da compra.
        #3. O caixa finaliza a compra.
        #4. O caixa informa o [valor total] da compra ao
        cliente.
        #5. O cliente informa que o [pagamento] será feito
```

```
        à vista e entrega o [dinheiro] ao caixa.
        #6. O caixa confere o valor entregue, verifica a
        necessidade de fornecer [troco] e finaliza a operação.
    </DESCRICAO>
</CENARIO_PRINCIPAL>
```

Cenários (Cursos) Alternativos

Listagem dos cenários em que as condições de uso da funcionalidade diferem do cenário principal. Cada cenário alternativo deve ser descrito como uma seqüência de passos para o cumprimento de uma tarefa por parte do usuário.

A listagem dos cenários alternativos estará delimitada pela tag <CENARIOS_ALTERNATIVOS>, e cada cenário alternativo é delimitado pela tag <CENARIO_ALTERNATIVO> e possui o seguinte formato:

- Título do cenário (<TITULO>): Descrição que sintetize em poucas palavras o conteúdo do cenário alternativo.
- Descrição (<DESCRICAO>): Texto que detalhe a seqüência de passos necessários para a realização do cenário. As entidades presentes no texto também devem estar denotadas entre colchetes ([Entidade]).

O ponto em que o cenário alternativo passa a diferir do cenário principal deve estar referenciado dentro da descrição do cenário principal. Isso facilita a leitura e a diferenciação dos cenários, que devem ter pontos em comum para serem considerados como cenários de um mesmo caso de uso. Um exemplo de como efetuar o vínculo entre os cenários pode ser observado no texto que se segue, no qual o cenário alternativo é referenciado no passo número cinco do cenário principal.

```
<CENARIO_PRINCIPAL>
    <TITULO> Compra de itens à vista </TITULO>
    <DESCRICAO>
        #1. O [cliente] entrega os [itens] da [compra] ao
        [caixa do supermercado].
        #2. O caixa passa [item] por item da compra.
        #3. O caixa finaliza a compra.
        #4. O caixa informa o [valor total] da compra ao
        cliente.
        #5. O cliente informa que o [pagamento] será feito
        à vista e entrega o [dinheiro] ao caixa (CA-01).
        #6. O caixa confere o valor entregue, verifica a
```

```
        necessidade de fornecer [troco] e finaliza a operação.
    </DESCRICAO>
</CENARIO_PRINCIPAL>

<CENARIOS_ALTERNATIVOS>
  <CENARIO_ALTERNATIVO>
    <TITULO> CA-01 - Compra de itens com cartão de crédito
    </TITULO>
    <DESCRICAO>
        #1. O [cliente] informa que o [pagamento] será
        feito com [cartão de crédito] e entrega o cartão
        ao [operador do caixa].
        #2. O caixa passa o cartão na [leitora], em seguida
        digita o [valor da compra].
        #3. O caixa aguarda a [autenticação] por parte da
        [central de cartão de crédito].
        #4. Com a [compra] autenticada, o atendente devolve
        o cartão ao cliente. Entrega agora o [comprovante da
        compra] para que o cliente assine-o.
        #5. O cliente assina o comprovante e o devolve ao
        operador do caixa.
        #6. O operador encerra a operação.
    </DESCRICAO>
  </CENARIO_ALTERNATIVO>
</CENARIOS_ALTERNATIVOS>
```

Exceções

As exceções descrevem eventuais falhas e/ou interrupções que podem ocorrer durante a execução de um cenário. Nesta seção serão listadas todas as exceções que podem ocorrer durante qualquer um dos cenários do caso de uso.

A listagem de exceções do UC é delimitada por uma tag <EXCECOES>. Já cada exceção deve ser delimitada pela tag <EXCECAO> e descrita como se segue:

- Título da exceção (<TITULO>): Descrição que sintetize em poucas palavras o conteúdo da exceção.
- Descrição (<DESCRICAO>): Texto que detalhe a exceção, apresentando o motivo da falha e as atitudes corretivas que o ator deve tomar.

As expressões lógicas que são testadas para detectar a ocorrência da exceção devem estar entre colchetes ([Expressão lógica]).

A exceção deve ser referenciada no corpo dos cenários em que a mesma pode ocorrer, por exemplo considere a seguinte exceção:

```
<EXCECOES>
  <EXCECAO>
    <TITULO> E-01 - Cartão de crédito não passou pela validação
    </TITULO>
    <DESCRICAO>
        [Se o prazo de validade do cartão de crédito tiver
        expirado], o sistema aborta a operação.
    </DESCRICAO>
  </EXCECAO>
</EXCECOES>
```

Veja como a mesma poderia ser referenciada dentro de um cenário:

```
<CENARIO PRINCIPAL>
    <TITULO> Compra de itens à vista </TITULO>
    <DESCRICAO>
        #1. O [cliente] entrega os [itens] da [compra] ao
        [caixa do supermercado].
        #2. O caixa passa [item] por item da compra.
        #3. O caixa finaliza a compra.
        #4. O caixa informa o [valor total] da compra ao
        cliente.
        #5. O cliente informa que o [pagamento] será feito
        com cartão de crédito (E-01).
        #6. O caixa fornece o comprovante ao cliente.
    </DESCRICAO>
</CENARIO PRINCIPAL>
```

Pós-Condições

Uma listagem das pós-condições que devem vigorar após o término do caso de uso é apresentada nesta seção, a listagem é delimitada pela tag <POS_CONDICOES>.

Cada pós-condição deve ser delimitada pela tag <POS_CONDICAO> e ser descrita da forma:

- Título: Texto com poucas palavras (no máximo 10) que sintetize a descrição da Pré-Condição (<TITULO>).
- Descritivo da pós-condição (<DESCRICAO>): Um texto explicando a pós-condição, seus objetivos e as particularidades que devem ser observadas. As entidades presentes no texto da pós-condicão devem estar entre colchetes ([Entidade]).

Observe a seguir um exemplo de definição de uma pós-condição:

```
<POS_CONDICOES>
    <POS_CONDICAO>
        <TITULO>
            POSC-01 - Retorno à tela de entrada do sistema
        </TITULO>
        <DESCRICAO>
            Após a execução de todas as funcionalidades, o
            sistema deve descartar as operações não salvas
            e retornar à [tela de entrada], onde podem ser
            preenchidas as informações de [login] do [usuário].
        </DESCRICAO>
    </POS_CONDICAO>
</POS_CONDICOES>
```

Referências a Outros UCS

Caso seja necessário referenciar um outro UC dentro do texto, é recomendável colocar a condição de ativação da chamada em um curso alternativo, assim a necessidade de interface entre os casos de uso acaba sendo medida (contabilizada).

6.1.4 Caso de Uso Completo

Nesta seção apresenta-se um exemplo de caso de uso utilizando o modelo e a notação em XML.

```
<CASO DE USO>
    <ATORES>
        <ATOR>
            <NOME> AT-01 - Cliente do banco </NOME>
            <DESCRICAO>
                Um [cliente] qualquer, registrado no banco
                e que deseja efetuar o [saque] através de um
```

```
            [terminal eletrônico].
        </DESCRICAO>
    </ATOR>
</ATORES>
<PRE_CONDICOES>
    <PRE_CONDICAO>
        <TITULO> PREC-01 - Cadastro do Cliente </TITULO>
        <DESCRICAO>
        O [cliente deve estar cadastrado como um dos correntistas no
        banco] que detém a agência.
        </DESCRICAO>
    </PRE_CONDICAO>
    <PRE_CONDICAO>
        <TITULO> PREC-02 - Disponibilidade de Saldo </TITULO>
        <DESCRICAO> [O cliente deve ter saldo disponível] </DESCRICAO>
    </PRE_CONDICAO>
    <PRE_CONDICAO>
        <TITULO> PREC-03 - Estado do caixa automático </TITULO>
        <DESCRICAO> [O caixa automático deve estar ligado] e
        [aguardando operação] </DESCRICAO>
    </PRE_CONDICAO>
</PRE_CONDICOES>
<CENARIO_PRINCIPAL>
    <TITULO> Saque de Dinheiro </TITULO>
    <DESCRICAO>
    #1. O [cliente] chega até o [caixa automático] desejando [sacar]
    uma [quantia em dinheiro].
    #2. O [sistema] do caixa automático está aguardando operação,
    emitindo uma [mensagem] que solicita ao cliente que insira seu
    [cartão do banco].
    #3. O cliente insere o cartão do banco [E-01], o sistema valida o
    cartão e solicita a [senha].
    #4. O cliente informa a sua senha [E-02].
    #5. O sistema valida a senha e fornece um [menu] com as seguintes
    opções: Consultar [saldo], Efetuar [saque], Sair.
    #6. O cliente seleciona Efetuar saque.[CA-01] [CA-02]
    #7. O sistema permite ao cliente a escolha do [valor a sacar].
    #8. O cliente informa o valor a sacar e confirma a operação.
    #9. O sistema verifica o [saldo], efetua o [saque], entrega o
    [dinheiro] e diminui o saldo do [cliente] conforme o [valor
    sacado].
    </DESCRICAO>
</CENARIO_PRINCIPAL>
<CENARIOS_ALTERNATIVOS>
    <CENARIO_ALTERNATIVO>
        <TITULO> CA-01 - Consultar Saldo </TITULO>
        <DESCRICAO>
        #1. O [cliente] escolhe [Consultar saldo].
        #2. O sistema emite um [comprovante] informando o [saldo do
```

```
        cliente].
        </DESCRICAO>
    </CENARIO_ALTERNATIVO>
    <CENARIO_ALTERNATIVO>
        <TITULO> CA-02 - Saída do sistema </TITULO>
        <DESCRICAO>
        #1. O [cliente] escolhe Sair
        #2. O sistema encerra a operação e devolve o [cartão] ao [cliente].
        </DESCRICAO>
    </CENARIO_ALTERNATIVO>
</CENARIOS_ALTERNATIVOS>
<EXCECOES>
    <EXCECAO>
        <TITULO> E-01 - Falha no cartão </TITULO>
        <DESCRICAO>
        1. O [cartão inserido não pertence ao banco] ou [está com
        problemas].
        2. O sistema emite mensagem informando o cliente sobre o
        problema ocorrido e volta a aguardar operação.
        </DESCRICAO>
    </EXCECAO>
    <EXCECAO>
        <TITULO> E-02 - Senha inválida </TITULO>
        <DESCRICAO>
        1. O [cliente informou uma senha inválida].
        2. O sistema emite mensagem informando o cliente sobre o
        problema ocorrido, solicita a senha novamente,até um máximo de
        três vezes.
        </DESCRICAO>
        </EXCECAO>
    <EXCECAO>
        <TITULO> E-03 - Saldo insuficiente </TITULO>
        <DESCRICAO>
        1. O [saldo é insuficiente para permitir o saque solicitado
        pelo cliente].
        2. O sistema informa ao cliente que o saldo (valor do saldo) é
        insuficiente para completar o saque e volta a solicitar a
        operação desejada (CP-Passo 5).
        </DESCRICAO>
    </EXCECAO>
</EXCECOES>
<POS_CONDICOES>
    <POS_CONDICAO>
        <TITULO> POSC-01 - Estado final do sistema </TITULO>
        <DESCRICAO>
            1. O sistema retornou ao [estado aguardando operação].
        </DESCRICAO>
    </POS_CONDICAO>
</POS_CONDICOES>
```

```
</CASO_DE_USO>
```

6.2 Protótipo

Visando à automatização da coleta do USP e USPF, um protótipo foi construído para efetuar os cálculos necessários. A ferramenta é bastante simples, eis algumas informações sobre a mesma:

- Interface Gráfica - GUI (Guided User Interface)
- Orientada a Objetos - Desenvolvida em Delphi.
- Permite o cálculo de qualquer UC armazenado na forma de um documento XML criado com a notação proposta.

A ferramenta avalia cada seção do UC, utilizando como subsídio as informações denotadas no texto das seções. Para cada seção do UC, a ferramenta adiciona uma tag que contém o número de USP e USPF de cada uma delas. Ao final um número de USP e USPF para o UC por completo é gerado.

6.3 Considerações Finais

A utilização de uma notação mais formal para escrever os UCs traz algumas vantagens, entre elas estão a padronização obtida (a documentação estará gerada em um mesmo formato) e as diversas possibilidades de automatização e utilização dos documentos por ferramentas.

Contudo, um dos motivos que tornou o modelo de UC tão popular, foi a liberdade e facilidade para se redigir UCs, a introdução de uma notação mais rígida limita as possibilidades do analista responsável pelos mesmos, além de dificultar a sua escrita.

A notação aqui descrita e o protótipo construído foram utilizados no estudo de caso que será apresentado no capítulo seguinte, o estudo foi realizado para avaliar as métricas propostas.

Capítulo 7

Estudo de Caso para Validação do USP e USPF

Nos capítulos anteriores foram realizadas algumas proposições:

- Métrica para UCs, o USP
- Métrica para UCs utilizando a teoria Fuzzy, o USPF
- Notação para escrita de UCs visando a sua automatização
- Protótipo para automatização da aplicação da métrica proposta.

Para validar e atestar a eficácia e viabilidade das propostas anteriormente descritas, foi realizado um estudo de caso com uma base de dados de um projeto real de uma instituição privada que atua no segmento de logística.

A base de dados é constituída de especificações realizadas com casos de uso, medidas de FP para os mesmos, histórico de produtividade da equipe e a avaliação de características do sistema desenvolvido.

O estudo reuniu 5 módulos do referido projeto, com cerca de 250 FP cada um, além do FP que já constava na base de dados, o estudo efetuará a aplicação do UCP e das duas métricas propostas, o USP e o USPF. São apresentados os resultados do estudo e algumas conclusões extraídas do mesmo.

7.1 Objetivos do Estudo de Caso

O objetivo final do estudo é validar a viabilidade de aplicação das métricas USP e USPF e compará-las com as técnicas mais populares e reconhecidas dentro da comunidade de desenvolvimento.

Objetivos secundários tratam da possibilidade de automatização da coleta e aplicação da métrica, que seria resultante da utilização da notação baseada em XML e do protótipo de uma ferramenta que coleta e efetua o cálculo da métrica.

7.2 Descrição

A composição do estudo, o projeto e a base de dados utilizada, seus componentes, a técnica utilizada e os passos realizados estão descritos nas próximas seções.

7.2.1 Elementos da Análise

Diversos elementos formam a base de conhecimento necessária à realização do estudo de caso. Os elementos estão assim organizados:

- Métricas de software
  - FPA - Análise de Pontos por Função;
  - UCP - Pontos por Caso de Uso;
  - USP - Pontos por Tamanho de Caso de Uso; e
  - USPF - Pontos por Tamanho de Caso de Uso Fuzzy;
- Base de dados de um projeto de logística
  - Especificação do sistema (casos de uso);
  - FP calculados para cada caso de uso;
  - Histórico interno de produtividade da equipe em horas/FP; e
  - Tabela de fatores de ajuste
- Notação para escrita de casos de uso em formato XML
- Protótipo de ferramenta para automação da obtenção da métrica

7.2.2 O Projeto

O projeto adotado foi escolhido por preencher os requisitos necessários para a condução deste estudo. Encontrar um projeto disponível para utilização com tais características foi um dos maiores desafios deste trabalho.

O projeto "DTR" (nome fictício que será utilizado para denominar o projeto no restante do estudo) é de um sistema de logística produzido por uma empresa privada de desenvolvimento de software, o sistema foi construído para atender às necessidades operacionais e de planejamento de uma empresa fornecedora de soluções em logística.

Decorreram aproximadamente 2 anos entre o início e a implantação do projeto completo, tempo em que foram desenvolvidas as atividades de especifição, construção, homologação e preparação para a implantação, requerendo aproximadamente 17000 horas de trabalho de profissionais especializados.

O processo de desenvolvimento adotado foi incremental e iterativo, sendo que a cada iteração eram disponibilizadas versões do produto para a empresa contratante. O processo formalizava e reunia muitas das etapas de desenvolvimento de um software, tais como:

- Análise de riscos de cada iteração;
- Especificação com casos de uso;
- Análise de pontos de função das funcionalidades presentes nos casos de uso;
- Estimativa de custos e prazos para o desenvolvimento;
- Validação das especificações por parte dos usuários;
- Sistematização do acompanhamento da atividade de testes; e
- Homologação por parte de equipe de homologação do cliente.

Devido a seu porte, o sistema foi contruído visando ao máximo a reutilização de código, extensibilidade e manutenibilidade do software, fatores que influenciaram diretamente na escolha da linguagem, metodologia e plataforma para o desenvolvimento.

Eis algumas características técnicas do produto:

- Ambiente distribuído (3 camadas)
- C++ como principal linguagem utilizada
- Banco de dados Oracle [34]
- Ferramenta de modelagem Rational Rose [38]
- Orientado a objetos

- Camada própria de persistência de objetos
- Plataforma Win32 [30]

7.2.3 Procedimento Adotado

A análise está estruturada em um conjunto de fases, organizadas seqüencialmente, o material necessário e o produto de cada fase estão apresentados a seguir.

Organização dos Módulos

Foram extraídos 5 módulos do sistema em questão, cada um correspondendo a aproximadamente um mês de trabalho da equipe e uma iteração do processo de desenvolvimento de software, o esforço realizado em cada mês é variável (em cada mês a quantidade de horas trabalhada foi diferente).

Uma característica importante dos dois primeiros módulos deve ser observada quando se analisa os resultados dos mesmos, esses módulos tiveram uma produtividade consideravelmente inferior aos demais desenvolvidos. Isto é devido à inexperiência da equipe com a tecnologia utilizada e as incertezas comumente encontradas no início de um projeto.

Divisão das Etapas

O experimento se divide em duas etapas bem distintas, a etapa de treinamento e a etapa de homologação ou validação.

A primeira etapa visa fornecer informações sobre a produtividade de cada métrica quando aplicada sobre o projeto em questão. A utilização de uma etapa de treinamento fornece dados de produtividade mais confiáveis que se os mesmos tivessem sido obtidos na literatura.

Para a etapa de treinamento foi escolhido um dos módulos do sistema, que foi extraído de uma etapa intermediária do desenvolvimento do software. Durante a fase de desenvolvimento do mesmo, a produtividade da equipe já estava há algum tempo bem estável, transformando-o em um bom candidato para o treinamento.

Ao final da etapa de treinamento, foram obtidos os valores de produtividade por hora de cada uma das métricas aplicadas, possibilitando a sua utilização na próxima etapa, a de validação.

Durante a etapa de validação, foram aplicadas as métricas sobre os 4 módulos restantes do conjunto de trabalho. Após a aplicação das métricas foram realizadas estimativas de esforço baseadas nas medidas de produtividades obtidas na etapa de treinamento.

Aplicação da Métrica UCP

A primeira métrica a ser aplicada sobre o projeto foi o UCP, para a coleta da métrica se fez necessária a avaliação dos atores e das entidades participantes do UC.

Os casos de uso originalmente construídos não continham explicitamente informações sobre os atores do UC, foi necessária uma leitura atenta do conteúdo dos mesmos para a identificação dos atores.

Os atores e os casos de uso foram mensurados manualmente, logo em seguida foram realizados os processos de ajuste de valor, baseando-se nas características técnicas do sistema e nas características da equipe e do processo de desenvolvimento adotado.

O ajuste foi realizado considerando-se as Tabelas 7.1 e 7.2. O cálculo dos fatores pode ser observado nas Equações 7.1 e 7.2.

Tabela 7.1: Peso dos Fatores Técnicos do Sistema DTR

Fator	Requisito	Influência	Peso	Total
F1	Sistema distribuído	4	2	8
F2	Tempo de resposta	4	2	8
F3	Eficiência	4	1	4
F4	Processamento complexo	3	1	3
F5	Código reusável	5	1	5
F6	Facilidade de instalação	0	0.5	0
F7	Facilidade de uso	0	0.5	0
F8	Portabilidade	0	2	0
F9	Facilidade de mudança	4	1	4
F10	Concorrência	5	1	5
F11	Recursos de segurança	3	1	3
F12	Acessível por terceiros	0	1	0
F13	Requer treinamento especial	2	1	2
Total				42

$$TCF = 0,6 + 0,01 * 42 = 1,02 \tag{7.1}$$

$$EF = 1,4 + (-0,03 * 29) = 0,53 \tag{7.2}$$

Após a classificação dos UCs, foi gerada uma predição do esforço necessário para a implementação das funcionalidades, a predição foi realizada utilizando os valores de produtividade fornecidos pela etapa de treinamento.

Tabela 7.2: Peso dos Fatores Ambientais do Sistema DTR

Fator	Descrição	Influência	Peso	Total
E1	Familiaridade com RUP ou outro processo formal	2	1.5	3
E2	Experiência com a aplicação em desenvolvimento	2	0.5	1
E3	Experiência em OO	3	1	3
E4	Presença de analista experiente	4	0.5	2
E5	Motivação	4	1	4
E6	Requisitos estáveis	4	2	8
E7	Desenvolvimento em meio-expediente	0	-1	0
E8	Linguagem de programação difícil	4	2	8
Total				29

Construção dos UCs em XML

Visando à automatização das próximas duas fases, os UCs do sistema são transformados em documentos XML.

Os UCs do sistema estavam construídos segundo uma notação própria da empresa que desenvolvia o software. Embora muito semelhante à notação proposta no Capítulo 6, os documentos estavam redigidos em formato Word[30] e precisavam, principalmente, ser transformados em documentos XML.

Algumas adaptações foram necessárias para essa geração, tais como:

- As informações passadas e recebidas pelos atores do caso de uso tiveram de ser contabilizadas;
- As entidades tiveram que ser identificadas e denotadas no texto;
- As expressões lógicas testadas nas prés-condições e nas exceções tiveram que ser identificadas e denotadas no texto; e
- Construção do documento que envolveu a geração de tags e a transposição do texto.

Ao final dessa etapa, os UCs necessários estavam escritos com a notação proposta e no formato de documentos XML.

Coleta da FPA

Conforme mencionado anteriormente, a FPA já havia sido aplicada sobre o projeto durante o seu desenvolvimento. Portanto, para obter os valores de FP por UC, foi necessário contabilizar as funcionalidades medidas (Entradas externas, Registros lógicos, etc), que se referiam a um mesmo UC.

Aplicação do USP

A primeira das métricas propostas a ser aplicada é o USP, duas tarefas importantes para a aplicação da mesma já foram realizadas, que foi a construção dos UCs em XML e a identificação (denotação) das entidades e expressões dentro do texto.

Dessa forma foi possível a utilização do protótipo construído para automatizar a coleta da métrica. Cada caso de uso foi medido separadamente através do mesmo, o qual informa o valor de USP total do UC e de cada uma de suas seções.

O ajuste foi realizado segundo as Tabelas 7.3 e 7.4, a obtenção do valor pode ser verificada pela Equação 7.3.

Tabela 7.3: Peso de Fatores Técnicos

Fator	Requisito	Influência
F1	Comunicação de dados	4
F2	Processamento Distribuído	4
F3	Desempenho	4
F4	Utilização do equipamento	3
F5	Volume de transações	1
F6	Entrada de dados on-line	5
F7	Eficiência do usuário final	3
F8	Atualização on-line	3
F9	Reutilização de código-fonte	3
F10	Processamento complexo	5
F11	Facilidade de implantação	0
F12	Facilidade Operacional	0
F13	Multiplicidade de locais	5
F14	Facilidade de mudanças	4
Total		45

Tabela 7.4: Peso de Fatores Ambientais

Fator	Descrição	Influência
E1	Existência de processo formal de desenvolvimento	2
E2	Experiência com a aplicação em desenvolvimento	0
E3	Experiência da equipe com as tecnologias utilizadas	0
E4	Presença de analista experiente	2
E5	Requisitos estáveis	2
Total		6

$$FA = 0.65 + (0.01 * 45) - (0.01 * 6) = 1.04 \quad (7.3)$$

As estimativas de esforço foram geradas com base nos valores de produtividade obtidos na etapa de treinamento.

Os resultados foram armazenados visando à comparação com as demais métricas aplicadas.

Aplicação do USPF

O último passo prático do estudo visa à aplicação do USPF. De forma muito similar ao USP, os passos necessários à aplicação automatizada do protótipo sobre os UCs construídos, estavam todos concluídos. As diferenças entre a métrica USP e USPF, residem agora, apenas na forma de cálculo da complexidade de cada uma das seções dos UCs, conhecimento este que está embutido no protótipo desenvolvido.

De forma muito similar à etapa de aplicação do USP, o próximo passo foi gerar as estimativas de esforço com base nos valores de produtividade obtidos na etapa de treinamento.

7.2.4 Resultados do Treinamento

O procedimento adotado para obter a produtividade através do módulo escolhido foi o seguinte:

1. Aplicam-se as métricas sobre o módulo;

   Os valores obtidos com a aplicação das métricas podem ser observados na Tabela 7.5.

Tabela 7.5: Valores Obtidos - Módulo de Treinamento

Métrica	UCP	FPA	USP	USPF
Total	11,35	66	113,36	117,52

2. Obtém-se a quantidade de horas gastas com o desenvolvimento;

   Quantidade de horas = 295,27

3. Divide-se o valor obtido com as métricas pelo no. de horas, o obtido está registrado na Tabela 7.6.

   Após uma breve análise dos valores de produtividade obtidos, podem ser destacados alguns pontos:

Tabela 7.6: Medidas de Produtividade (Esforço)

Medida	Valor
FP/Hora	0,2235
UCP/Hora	0,0384
USP/Hora	0,3839
USPF/Hora	0,3979

- O valor de produtividade obtido para a FPA, foi de 0,2235 FP/h, ou seja, aproximadamente 4,5 h/FP.
- O valor de produtividade sugerido por Karner para utilização do UCP e de 20 h/UCP [22]. Transformando-se a produtividade de 0,0384 UCP/h, obtém-se o valor de 26 h/UCP, valor próximo ao sugerido por Karner.

7.2.5 Resultados da Aplicação das Métricas

Finalmente, após a aplicação das métricas, os resultados encontram-se dispostos nas Tabelas 7.7, 7.8, 7.9 e 7.10.

Tabela 7.7: Valores Obtidos - Módulo 2

UC	UUCP	UCP	UFP	FP	UUSP	USP	UUSPF	USPF
A2-UC01	-	-	17	18,7	42	43,68	42	43,68
A2-UC02	-	-	11	12,1	17	17,68	17	17,68
A2-UC03	-	-	6	6,6	20	20,8	20,4	21,21
A2-UC04	-	-	22	24,2	20	20,8	22	22,88
A2-UC05	-	-	22	24,2	30	31,2	32	33,28
A2-UC06	-	-	22	24,2	20	20,8	22,4	23,29
Total	20	10,81	100	110	149	154,96	155,8	162,03

Antes de uma avaliação dos resultados obtidos, cabe avaliar a influência exercida pela etapa de ajuste de cada métrica. Veja o peso exercido pelos fatores em cada uma das métricas:

- FPA - 1,1
- UCP - 0,54
- USP e USPF - 1,04

Tabela 7.8: Valores Obtidos - Módulo 3

UC	UUCP	UCP	UFP	FP	UUSP	USP	UUSPF	USPF
A3-UC07	-	-	6	6,6	15	15,6	15	15,6
A3-UC08	-	-	6	6,6	26	27,04	26	27,04
A3-UC09	-	-	41	45,1	79	82,16	83	86,32
A3-UC10	-	-	22	24,2	28	29,12	29,2	30,36
A3-UC11	-	-	32	35,2	55	57,2	61,8	64,27
A3-UC12	-	-	37	40,7	87	90,48	91,8	95,47
Total	25	10,81	144	158,4	290	301,6	306,8	319,06

Tabela 7.9: Valores Obtidos - Módulo 4

UC	UUCP	UCP	UFP	FP	UUSP	USP	UUSPF	USPF
A4-UC13	-	-	47	51,7	61	63,44	64,6	67,18
A4-UC14	-	-	45	49,5	66	68,64	70	72,8
A4-UC15	-	-	44	48,4	85	88,4	87,8	91,31
A4-UC16	-	-	28	30,8	44	45,76	44,8	46,59
A4-UC17	-	-	21	23,1	47	48,88	50,2	52,20
A4-UC18	-	-	33	36,3	37	38,48	39	40,56
Total	26	14,05	218	239,8	340	353,6	356,4	370,64

Tabela 7.10: Valores Obtidos - Módulo 5

UC	UUCP	UCP	UFP	FP	UUSP	USP	UUSPF	USPF
A5-UC19	-	-	40	44	48	49,92	48,4	50,33
A5-UC20	-	-	61	67,1	85	88,4	85	88,4
A5-UC21	-	-	59	64,9	81	84,24	82,2	85,48
A5-UC22	-	-	33	36,3	53	55,12	57	59,28
A5-UC23	-	-	36	39,6	65	67,6	68,6	71,34
Total	25	13,51	229	251,9	332	345,28	341,2	354,84

A etapa de ajuste da FPA é composta unicamente pela aplicação dos fatores de ajuste técnicos, que elevam os valores obtidos.

O ajuste na técnica de UCP inclui além do ajuste dos fatores técnicos o ajuste proporcionado pelos fatores ambientais que, em geral, reduzem o valor bruto obtido. Porém observa-se na fórmula do ajuste do UCP, que os fatores ambientais possuem maior peso que os fatores técnicos, isso acaba fazendo com que o valor geral seja reduzido após o ajuste, e não ampliado como ocorre na FPA.

O ajuste através dos fatores ambientais também está presente no USP e USPF, porém de uma forma diferente que o existente na técnica UCP, no USP os fatores ambientais são em menor número e tem o mesmo peso que os fatores técnicos, dessa forma a tendência é de que o ajuste eleve o valor bruto, porém de uma forma um pouco mais suave do que na FPA.

7.2.6 Estimativas de Esforço

Com a produtividade fornecida pela etapa de treinamento e com os valores das métricas obtidos, o próximo passo é estimar o esforço e observar como se comportam as métricas com relação ao efetivamente obtido durante o desenvolvimento do sistema.

O esforço de cada módulo é estimado da seguinte forma:

Esforço = Tamanho do Módulo (em FP, UCP, USP, USPF) / Produtividade

O resultado da predição de esforço pode ser observado na Tabela 7.11.

Tabela 7.11: Estimativas de Esforço em Horas

Módulo	UCP	FP	USP	USPF	Realizado
Módulo A2	281,21	492,17	403,63	407,11	1258
Módulo A3	351,52	708,72	785,60	801,69	1051
Módulo A4	676,25	1072,93	921,04	931,29	1111
Módulo A5	351,52	1230,42	899,37	891,58	1007

Tabela 7.12: Taxas de Erro das Estimativas

Módulo	UCP	FPA	USP	USPF
Módulo A2	77,6%	60,8%	67,9%	67,3%
Módulo A3	66,5%	32,5%	25,2%	23,7%
Módulo A4	39,1%	3,4%	17%	16,1%
Módulo A5	65,1%	22,1%	10,74%	11,51%

A Figura 7.1 apresenta as taxas de erro obtidas comparando-se o estimado pela métrica com o efetivamente realizado. As porcentagens estão na Tabela 7.12.

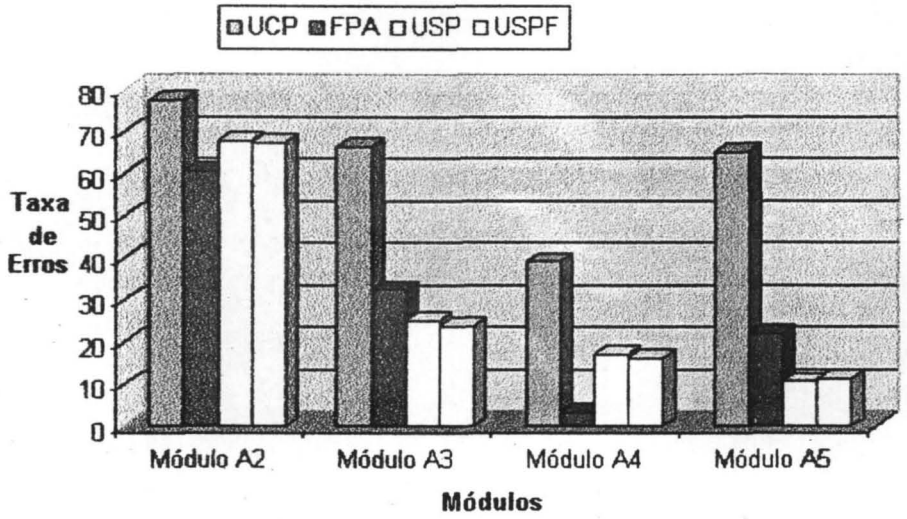

Figura 7.1: Taxas de Erro das Estimativas

7.2.7 Análise dos Resultados

UCP x USP

Conforme pode ser observado na Tabela 7.12, o UCP apresentou as maiores taxas de erro em todos os módulos analisados. Em parte, isto é devido ao ajuste aplicado, porém ainda que os fatores de ajuste ambientais não tivessem sido aplicados, os valores de desvio em relação ao realizado seriam os maiores em relação às demais métricas.

Os resultados vêm confirmar algumas das suspeitas colocadas no início deste trabalho, onde se comentava que a granularidade e o tamanho dos UCs poderiam influenciar os resultados apresentados pelo UCP. O USP embora tome como base também os UCs, apresentou resultados mais consistentes, com margens de erro bem inferiores em todos os módulos analisados.

Módulos Iniciais

Os resultados apresentados pelas Tabelas 7.11 e 7.12 mostram que as estimativas geradas por todas as métricas nos dois primeiros módulos (A2 e A3)

tiveram resultados piores quando comparadas às estimativas geradas para os dois módulos subseqüentes (A4 e A5).

A inexperiência da equipe com a metodologia, área de negócio e com o ferramental utilizado e a instabilidade dos requisitos no início de um projeto contribui para que a produtividade seja consideravalmente inferior nos primeiros estágios do projeto. Dessa forma, ao tentarmos realizar estimativas para os primeiros estágios tendo como base uma produtividade média ou uma produtividade gerada a partir de um treinamento realizado sobre um estágio posterior, as estimativas acabam sendo sub-estimadas, provocando as maiores diferenças entre o estimado e o realizado.

FPA x USP

As margens de erro apresentadas pela FPA e pelo USP/USPF estão razoavelmente próximas, com uma técnica apresentando melhores resultados que a outra em um ou outro módulo. Isso corrobora a idéia de proximidade existente entre as funcionalidades existentes no caso de uso (utilizada pela FPA) e o número de seções, entidades, expressões também presentes no mesmo (utilizado pelo USP/USPF).

Como observação cabe registrar que o ótimo desempenho da FPA no módulo A4, deve-se ao fato da produtividade registrada neste módulo ter sido muito próxima à produtividade obtida no módulo de treinamento, o que fez com que as estimativas geradas ficassem muito próximas aos valores reais.

USP x USPF

O USPF apresentou melhores resultados que o USP, conforme observa-se na Tabela 7.12, apesar das diferenças existentes entre as margens de erro terem sido muito pequenas.

A proximidade entre as duas métricas deve-se em parte às tabelas de classificação das seções do USP e USPF, em tais tabelas observa-se uma proximidade grande entre as faixas de complexidade, por exemplo veja a Tabela 7.13.

Tabela 7.13: Complexidade das Exceções

Complexidade	No. expressões testadas	Qtde. USP
Simples	1	1
Média	2 ou 3	2
Complexa	> 3	3

A diferença entre as faixas de complexidade é de apenas 1 ou 2 pontos

(expressões lógicas); quando estendido na forma de um *número Fuzzy*, essa classificação acaba não apresentando regiões de transição entre as faixas de complexidade, limitando a atuação da *teoria Fuzzy* sobre a métrica.

Futuras extensões da métrica que modifiquem as tabelas de classificação, ampliando o número de elementos considerados ou ampliando as diferenças entre as faixas poderão usufruir mais da classificação gradual proporcionada pelo Fuzzy.

7.2.8 Considerações Finais

Este capítulo apresentou os resultados extraídos do estudo de caso realizado para validação das métricas propostas e também para avaliar as possibilidades de automatização.

Os resultados obtidos demonstram que as estimativas geradas com as métricas propostas, USP e USPF, apresentam capacidade de predição muito similar àquelas geradas com a técnica mais popular, a FPA.

As margens de erro obtidas com a aplicação do UCP corroboram a hipótese de que a técnica possui limitações para quantificar a funcionalidade presente nos UCs, pois apresentou taxas de erro muito superiores àquelas apresentadas pelas demais métricas.

Os resultados demonstrados pelo USPF, embora muito similares aos apresentados pelo USP, demonstram mais uma vez as possibilidades de aplicação da teoria *Fuzzy* no contexto do processo de medição de software. A extensão e modificação das tabelas de classificação e de suas respectivas faixas de complexidade, poderá ampliar os ganhos obtidos com a aplicação do *Fuzzy* sobre as mesmas.

A automatização da coleta através do uso de uma notação estruturada e de uma ferramenta permitiu maior agilidade e confiabilidade no processo de medição. No entanto, como o protótipo não dava condições para facilitar a geração dos UCs na notação proposta, as dificuldades para adaptá-los à notação, foram consideráveis.

O capítulo seguinte, traz as conclusões finais sobre as propostas, estudos e experimentos realizados neste trabalho.

Capítulo 8

Conclusão

Conforme exposto neste trabalho, a especificação realizada com casos de uso é a maneira mais utilizada e efetiva de se documentar requisitos em sistemas orientados a objeto. As métricas atualmente existentes impõem dificuldades e algumas restrições no processo de medição de software através de casos de uso.

O efetivo sucesso de uma métrica depende não apenas de sua capacidade técnica, mas também da facilidade de utilização e da possibilidade de automatização.

O modelo UCP vem se popularizando entre a comunidade de desenvolvimento de software devido ao mesmo ser bastante simples e especificamente projetado para a utilização com casos de uso. Contudo, o modelo apresenta ainda algumas deficiências na mensuração, o que pode provocar distorções em estimativas geradas com base no modelo, conforme constatado no estudo de caso realizado.

A métrica apresentada, o USP, supera algumas das limitações existentes no UCP sem se afastar do modelo de UCs, mantendo a simplicidade no processo de medição. Os resultados do estudo de caso demonstram que as estimativas geradas com o USP são mais precisas e confiáveis que aquelas geradas com o UCP. A introdução dos conceitos *Fuzzy* no processo, através da métrica USPF, melhora as capacidades de dimensionamento das tabelas de classificação, conforme já fora constatado em outras extensões que também utilizaram a teoria *Fuzzy* [21] neste contexto.

O estudo de caso foi realizado sobre uma base de dados de 5 módulos de um projeto de informatização de uma empresa de logística. Todos os UCs do projeto foram convertidos para a notação proposta em XML, esse processo foi bastante trabalhoso, já que os UCs estavam construídos com outra notação e formato de arquivo. Após a conversão, a aplicação da métrica através do protótipo construído foi trivial, evidenciando a necessidade de utilização de

uma notação estruturada visando à automatização da aplicação da métrica.

Os resultados apresentados pelo estudo demonstram que as estimativas geradas pelo USP e pelo USPF têm capacidade preditiva similar à FPA e, com taxas de erro muito inferiores às apresentadas pelo UCP.

8.1 Trabalhos Futuros

Sugere-se para trabalhos futuros a extensão dos seguintes tópicos:

- Tabelas de Classificação: As tabelas que classificam as seções do UC poderão ser estendidas de forma a considerar outros itens presentes no UC, visando a uma melhor mensuração do mesmo. Como exemplo, pode-se citar a possibilidade de se construir uma matriz para classificação dos cenários (a exemplo do que ocorre no FP), onde seriam consideradas as entidades e seus respectivos atributos.

- Fuzzy: No caso da criação destas matrizes, também seria interessante a aplicação do *Fuzzy* para a geração de classificações graduais. Além disso, um estudo de caso poderia investigar e comparar a capacidade e adequação dos diferentes números fuzzy ao processo de medição, por exemplo poderia ser realizada uma comparação entre os números triangulares e os trapezoidais. O processo de defuzificação poderia também ser transformado para que métodos clássicos de defuzificação sejam utilizados (como os citados na Seção 3.6).

- Notação: Nesse trabalho, propôs-se uma notação XML para facilitar a automatização e coleta das métricas. Entretanto, em [25] é apresentada uma outra notação, utilizando um formato específico para a descrição de metadados o XMI (XML Metadata Interchange-XMI [47]). O XMI já é utilizado em outras ferramentas de modelagem como o Together [5] e deverá também ser explorado em trabalhos futuros de automatização das métricas USP e USPF.

- Protótipo: O protótipo construído poderia ser estendido para permitir a aplicação automática de outras métricas e ainda para permitir a construção do UC diretamente no protótipo, eliminando a necessidade de se construir o mesmo em uma notação mais formal como o XML ou XMI (o protótipo poderia fazer a geração automática do UC nesta notação).

- **Estudos de Caso:** Novos estudos de caso com outros tipos de projeto e técnicas de escrita de UCs poderiam ser realizados para identificar deficiências e promover o refinamento das métricas propostas.

Referências Bibliográficas

[1] M. Aguiar. PF ou UCP? Como Estimar Projetos Orientados a Objetos. *Developers' Magazine*, January 2003.

[2] A. Albrecht. Measuring Application Development Productivity. In *Proc. of the IBM Applications Development Symposium*, pages 83–92, October 1979.

[3] G. Banerjee. Use Case Points - An Estimation Approach. Available from `http://undergraduate.cs.uwa.edu.au/units/670.300/readings/usecasepoints.pdf`, April, 2004, August 2001.

[4] Borland® CaliberRM. Available from `http://www.borland.com.br/caliber/index.html`, April, 2004.

[5] Borland® Together. Available from `http://www.borland.com.br/together/index.html,April,2004`.

[6] M. R. Braz. Tecnologia de Web Services - Definições e Perspectivas. *Developers' Magazine*, pages 22–24, April 2003.

[7] G. Caldiera, G. Antoniol, R. Fiutem, and C. Lokan. Definition and Experimental Evaluation of Function Points for Object-Oriented Systems. In *Proceedings of the 5th. International Symposium on Software Metrics*, pages 167–178, March 1998.

[8] Cetus Links - Object-Orientation. Available from `http://www.cetus-links.org/oo_xml.html`, April, 2004.

[9] A. Cockburn. *Writing Effective Use Cases.* Addison-Wesley, 2001.

[10] Cost Xpert. Available from `http://www.costxpert.com`, April, 2004.

[11] Costar. Available from `http://www.softstarsystems.com`, April, 2004.

[12] Enterprise Architect - UML design tool. Available from `http://www.sparxsystems.com.au/ea.htm`, April, 2004.

[13] Estimate Easy Use Case. Available from `http://www.duvessa.com/products.html`, April, 2004.

[14] Extensible Markup Language (XML). Available from `http://www.w3c.org/xml`, April, 2004.

[15] T. Fetcke, A. Abran, and T. Nguyen. Mapping the OO-Jacobson Approach into Function Point Analysis. *Technology of Object-Oriented Languages and Systems*, pages 192–202, July-August 1997.

[16] E. Freire. O Método de Pontos de Caso de Uso e o Cálculo de Estimativas. *Developers' Magazine*, February 2003. In Portuguese.

[17] IBM Rational® RequisitePro®. Available from `http://www-306.ibm.com/software/awdtools/reqpro`, April, 2004.

[18] IBM Rational Rose Developer®. Available from `http://www-306.ibm.com/software/awdtools/developer/rose/`, April, 2004.

[19] International Function Point Users Group (IFPUG). *Function Point Counting Practices Manual*, 4.1.1 edition, April 2000. Ordering available from `http://www.ifpug.org/publications/manual.htm`, April, 2004.

[20] I. Jacobson, M. Christerson, P., and G. Övergaard. *Object Oriented Software Engineering: A Use-Case Driven Approach.* Addison-Wesley, 1992.

[21] O. S. L. Junior. Análise de Pontos por Função Fuzzy. Masterthesis, Universidade de Fortaleza, 2002. Available from `http://www.unifor.br/hp/pos/mia/13.zip`, April, 2004.

[22] G. Karner. Use Case Points - Resource Estimation for Objectory Projects, 1993. Copyright owned by Rational software, referenced by [3, 15].

[23] G. Klir and T. Folger. *Fuzzy Sets, Uncertainty and Information.* Prentice-Hall, 1998.

[24] S. Kusumoto, K. Inoue, T. Kasimoto, A. Suzuki, K. Yuura, and M. Tsuda. Function Point Measurement for Object-Oriented Requirements Specification. In *The 24th Annual International Computer Software and Applications Conference, COMPSAC 2000*, pages 543 – 548, October 2000.

[25] S. Kusumoto, F. Matukawa, K. Inoue, S. Hanabusa, and Y. Maegawa. Effort Estimation Tool Based on Use Case Points Method. Submitted to METRICS2004, available from `http://sel.ics.es.osaka-u.ac.jp/~lab-db/betuzuri/archive/462/462.pdf`, April, 2004, February 2004.

[26] C. Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process.* Prentice-Hall, second edition, 2002.

[27] M. Ma, A. Kandel, and M. Friedman. A New Approach for Defuzzification. *Fuzzy Sets and Systems*, pages 351–356, 2000.

[28] M. Marchesi. OOA Metrics for the Unified Modeling Language. In *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering (CSMR'98)*, page 67. IEEE Computer Society, March 1998.

[29] O. Mendes, A. Abran, and P. Bourque. An FP Tool Classification Framework and Market Survey, 1996. Available from `http://www.lrgl.uqam.ca/publications/pdf/83.pdf`, April, 2004.

[30] Microsoft Corporation. Available from `http://www.microsoft.com`, April, 2004.

[31] S. Nageswaran. Test Effort Estimation Using Use Case Points (UCP). In *14th International Software / Internet Quality Week*, May-June 2001. Available from `http://www.cognizant.com/cogcommunity/presentations/Test_Effort_Estimat%ion.pdf`.

[32] V. Novak and I. Perfilieva. Evaluating Linguistic Expressions and Functional Fuzzy Theories in Fuzzy Logic. In *Computing with Words in Information/Intelligent Systems*, pages 383–406, Berlin, 1999. Springer-Verlag.

[33] Object Management Group (OMG). *Unified Modeling Language Specification*, 1.4 edition, September 2001. Available from `http://www.omg.org/cgi-bin/doc?formal/01-09-67`, April, 2004.

[34] Oracle Corporation. Available from `http://www.oracle.com`, April, 2004.

[35] W. Pedrycz and F. Gomide. *An Introduction to Fuzzy Sets - Analysis and Design.* The MIT Press, 1998.

[36] R. S. Pressman. *Software Engineering.* McGraw-Hill, fifth edition, 2002.

[37] J. Ram and S. Raju. Object Oriented Design Function Points. *IEEE Transactions on Software Engineering*, pages 121–126, October 2000.

[38] Rational Software. Available from `http://www.rational.com`, April, 2003.

[39] E. Schimitz and D. Silveira. *Desenvolvimento de Software Orientado a Objetos.* Brasport, 2000.

[40] I. Sommerville. *Engenharia de Software.* Addison-Wesley, sixth edition, 2003.

[41] Sun Microsystems. Available from `http://www.sun.com`, April, 2004.

[42] C. Symons. *Software Sizing and Estimating: MKII Function Point Analysis.* Wiley and Sons, 1991.

[43] T. Uemura, S. Kusumoto, and K. Inoue. Function Point Measurement Tool for UML Design Specification. In *Sixth IEEE International Symposium on Software Metrics*, pages 62–69, November 1999.

[44] United Kingdom Software Metrics Association (UKSMA). *MKII Function Point Analysis Counting Practices Manual*, 1.3.1 edition, September 1998. Available from `www.uksma.co.uk/public/mkIIr131.pdf`, April, 2004.

[45] W. Van Leekwijck and E. E. Kerre. Defuzzification: criteria and classification. *Fuzzy Sets and Systems*, 108(2):159–178, December 1999.

[46] P. Vickers. *An Introduction to Function Point Analysis.* School of Computing and Mathematical Sciences, Liverpool John Moores University, 1998. Available from `http://computing.unn.ac.uk/staff/cgpv1/downloadables/fpa.pdf`, April, 2004.

[47] XML Metadata Interchange (XMI) specification. Available from `http://www.omg.org/docs/formal/03-05-02.pdf`, April, 2004.

[48] C. Yau and R. Tsoi. Assessing the Fuzziness of General System Characteristics in Estimating Software Size. *IEEE Transactions on Software Engineering*, pages 189–193, 1994.

[49] L. A. Zadeh. Fuzzy sets. *Information and Control*, 3(8):338–353, June 1965.

Apêndice A

Glossário

- AFPC: Adjusted Function Point Calculation (Cálculo de Pontos por Função Ajustado)
- API: Aplication Program Interface (Interface de programação de aplicação)
- BFPUG: Brazilian Function Point Users Group (Grupo Brasileiro de Usuários de Pontos por Função)
- CA: Complexidade do ator
- CE: Complexidade da exceção
- COCOMO: Constructive Cost Model (Modelo de Custo Construtivo)
- CPoC: Complexidade da pós-condição
- CPrC: Complexidade da pré-condição
- DET: Data Element Type (Item de dados referenciado)
- EBP: Elementary Business Process (Processo Elementar de Negócio)
- EED: Elemento de Entrada de Dados
- EF: Environmental Factor (Fator de Ajuste Ambiental)
- EI: External Input (Entrada externa do usuário)
- EIF: External Interface File (Arquivo de Interface Externa)
- EO: External Output (Saída do Usuário)
- EQ: External Inquiry (Consulta do Usuário)

- ER: Entidade referenciada
- ESD: Elemento de Saída de Dados
- FAA: Fator Ambiental de Ajuste
- FFPA: *Fuzzy* Function Points Analysis (Análise de Pontos por Função *Fuzzy*)
- FP: Function Points (Pontos por Função)
- FPA: Function Points Analysis (Análise de Pontos por Função)
- FTA: Fator Técnico de Ajuste
- FTR: File Type Referenced (Arquivo referenciado)
- GUI: Graphical User Interface (Interface Gráfica com o Usuário)
- HTML: Hypertext Markup Language (Linguagem de Marcação de Hipertexto)
- IFPUG: International Function Point Users Group (Grupo Internacional de Usuários de Pontos por Função)
- ILF: Internal Logical File (Arquivo Lógico Interno)
- LOC: Lines of Code (Linhas de Código)
- MKII FPA: Mark II Function Points Analysis (Análise de Pontos por Função Mark II)
- OO: Object Oriented/Object Orientation (Orientado a Objetos/Orientação a Objetos)
- OODFP: Object Oriented Design Function Points (Pontos por Função de Projeto Orientado a Objetos)
- PCA: Complexidade do cenário alternativo
- PCP: Complexidade do cenário principal
- RET: Record Element Types (Registro Lógico)
- SGBD: Sistema Gerenciador de Banco de Dados
- SGML: Standard Generalized Markup Language (Linguagem Padrão de Marcação Generalizada)

- TCA: Technical Complexity Adjustment (Ajuste de Complexidade Técnica)
- TCF: Technical Complexity Factor (Fator de Ajuste Técnico)
- TPA: Total de pontos da seção de atores
- TPCA: Total de pontos da seção de cenários alternativos
- TPE: Total de pontos da seção de exceções
- TPPoC: Total de pontos da seção de pós-condições
- TPPrC: Total de pontos da seção de pré-condições
- UAW: Unajusted Actor Weight (Peso não ajustado dos atores)
- UC: Use Case (Caso de Uso)
- UCP: Use Case Points (Pontos por Caso de Uso)
- UFP: Unajusted Function Points (Pontos por Função não ajustados)
- UML: Unified Modeling Language (Linguagem de Modelagem Unificada)
- USP: Use case Size Points (Pontos por Tamanho de Caso de Uso)
- USPF: *Fuzzy* Use case Size Points (Pontos por Tamanho de Caso de Uso *Fuzzy*)
- UUCP: Unajusted Use Case Points (Pontos não ajustados por Caso de Uso)
- UUCW: Unajusted Use Case Weight (Peso não ajustado de Casos de Uso)
- UUSP: Unajusted Use case Size Points (Pontos não ajustados por Tamanho de Caso de Uso)
- UUSPF: Unajusted *Fuzzy* Use case Size Points (Pontos Não Ajustados por Tamanho de Caso de Uso *Fuzzy*)
- VAF: Value Adjustment Factor (Fator de Ajuste de Valor)
- XMI: XML Metadata Interchange (XML para Intercâmbio de Metadados)
- XML: eXtended Markup Language (Linguagem de Marcação Extendida)